A Report on

Premium Customer Information Management

SUBBMITTED By

BHARATH NARAHARI
ID # 0728344

SUBMITTED
TO

Prof. Gonhsin Liu

Table of Contents

List of Tables Used:

1. CompanyInfo_M_T (Company Information Master Table)
2. EmpCategory_M_T (Employee Category Master Table)
3. Employee_M_T (Employee Master Table)
4. User_M_T (User Master Table)
5. UserPrivileges_M_T (User Privileges Master Table)
6. VehicleType_M_T (Vehicle Type Master Table)
7. VehicleManufacturer_M_T (Vehicle Manufacturer Master Table)
8. Vehicle_M_T (Vehicle Master Table)
9. Vendor_M_T (Vendor Master Table)
10. Spares_M_T (Spares Master Table)
11. Service_M_T (Service Master Table)
12. SparesInward_M_T (Spares Inward Master Table)
13. SparesInward_T_T (Spares Inward Transaction Table)
14. SparesOutward_M_T (Spares Outward Master Table)
15. SparesOutward_T_T (Spares Outward Transaction Table)
16. SparesInv_T_T (Spares Inventory Table)
17. Customer_M_T (Customer Master Table)
18. Enquiry_M_T (Enquiry Master Table)
19. Appointments_M_T (Appointments Master Table)
20. Quotation_M_T (Quotation Master Table)
21. QuotationService_T_T (Quotation Service Transaction Table)
22. QuotationSpares_T_T (Quotation Spares Transaction Table)
23. JobCard_M_T (Job Card Master Table)
24. JobCard_Emp_T_T (Job Card Employee Transaction Table)
25. Invoice_M_T (Invoice Master Table)
26. InvoiceService_T_T (Invoice Service Transaction Table)
27. InvoiceSpares_T_T (Invoice Spares Transaction Table)
28. UOM_M_T (Unit of Measurement Master Table)

Screens:

1. Company Information Screen
2. Employee Category Entry Screen
3. Employee Entry Screen
4. User Entry Screen
5. User Privileges Entry Screen
6. Vehicle Type Entry Screen
7. Vehicle Manufacturer Entry Screen
8. Vehicle Entry Screen
9. Vendor Entry Screen
10. Unit of Measurement Entry Screen
11. Spares Entry Screen
12. Service Entry Screen
13. Spares Inward Entry Screen
14. Spares Outward Entry Screen
15. Customer Entry Screen
16. Enquiry Entry Screen
17. Appointment Entry Screen
18. Quotation Entry Screen
19. Job Card Entry Screen
20. Invoice Entry Screen
21. Login Screen

Reports:

1. Technicians Availability Report

2. Quotation
    a) Quotation Bill wise
    b) Quotations Consolidated Between Dates
    c) Quotations Approved
    d) Quotations Rejected / Cancelled
    e) Pending Quotations
3. Job Card
    a) Job Card for Each Employee (Online Bill)
    b) Employee wise Job Cards Between Dates

4. Invoice
    a) Invoice Report Bill wise
    b) Consolidated Invoices between Dates
    c) Vehicle wise Billing Report Between Dates
    d) Manufacturer wise Billing Report Between Dates

5. Inventory
    a) Spares Inward Report Bill wise
    b) Spares Inward Between Dates
    c) Spares Inward Item wise
    d) Spares Outward Report Bill wise
    e) Spares Outward Between Dates
    f) Spares Outwards Item wise
    g) Inventory Status Report Current Balance
    h) Inventory Status Report Between Dates
    i) ROL Report of Spares
    j) Group wise Spares List
6. Vendors
    a) List of Vendors
    b) List of Vendors for Selected Item
    c) Vendor wise Purchases between dates

ABSTRACT

The system **Premium Customer Information Management** is a client-server application designed by keeping in view the various customer invoices that are maintained at automobile service center. An automobile service center which deals with hundreds of vehicles daily where in people need a system which quickly provide services which could enhance the day to day activities of the service center with efficiency and correctness.

This system **Premium Customer Information Management** is a user friendly, GUI based client/server application that will automate the different activities involved in the day to day activities of the QSC (Quick Service Center).

An application for completely automating the different activities of the various departments or sections and maintain the status of different processes. The system will help in reducing the time and effort in preparation quotation, invoice and reports.

The company (Quick Service Center) provides services to the vehicle owners of whose vehicles may fall under any category of vehicles like two wheelers, Four Wheelers, 3 stroke, 4 Stroke, Etc., the Quick Service Center Receives calls/ inquiries from the vehicle owners who generally ask for the details pertaining to the various services provided and the charges for various services and spares.

The company purchases spare parts from different to meet the requirements of its customers. After having the information regarding the various services and charges, customer comes to the Quick Service Centers with his vehicle where the condition of the vehicle is tested by the technician and he gives the details of required services and spares that need to be used to repair the vehicle.

Authorized employee of Customer Care Division prepares the quotation and forwards it to the customer for its approval. Approval of the Quotation is being collected and the Job Card containing the Details of technicians who will be working on the job will be recorded and distributed to the respective technicians for their reference. Once the Vehicle is sent to the Repairs Section, Technicians will collect the required spares from the inventory and repair the vehicle. After the Job is completed, Invoice is raised for the spares used and services provided including tax applicable and collect the amount from the customer.

HARDWARE REQUIREMENTS:-

- Pentium III Min. 700 MHz.
- 128 MB RAM
- 512 KB Cache Memory
- Hard disk 20 GB

- Microsoft Compatible 101 or more Key Board

SOFTWARE REQUIREMENTS: -

Operating System	:	Windows 2000
Back End	:	Oracle8I or SQL Server 2000
Technologies	:	VB.NET, ADO.NET

Introduction

1.1 Existing System

The company (Quick Service Center) provides services to the vehicle owners of whose vehicles may fall under any category of vehicles like two wheelers, Four Wheelers, 3 stroke, 4 Stroke, Etc., the Quick Service Center Receives calls/ inquiries from the vehicle owners who generally ask for the details pertaining to the various services provided and the charges for various services and spares. The company purchases spare parts from different to meet the requirements of its customers. After having the information regarding the various services and charges, customer comes to the Quick Service Centers with his vehicle where the condition of the vehicle is tested by the technician and he gives the details of required services and spares that need to be used to repair the vehicle. Authorized employee of Customer Care Division prepares the quotation and forwards it to the customer for its approval. Approval of the Quotation is being collected and the Job Card containing the Details of technicians who will be working on the job will be recorded and distributed to the respective technicians for their reference.

Once the Vehicle is sent to the Repairs Section, Technicians will collect the required spares from the inventory and repair the vehicle.
After the Job is completed, Invoice is raised for the spares used and services provided including tax applicable and collect the amount from the customer.

1.2 Drawbacks of the existing System

1. The QSC is maintaining the records manually in the form of register entries.
2. The QCS was not in a position to monitor the different phase's activities.
3. Status of each and every document should be maintained which involves lot of manual process.
4. Reports for analysis were not available, readily.

System Analysis

2.1 Problem Definition

The first step is to define the problem that led to the Client's Request. It must also state the Objectives the Client is expecting to provide and the Benefits the Client wants to see.

2.1.1. Definition:

To develop an application for completely automating the different activities of the various departments or sections and maintain the status of different processes.

2.1.2. Objectives:

- To reduce the time, effort and errors in the preparation quotation, invoice, and so on.
- To provide various analysis reports.
- To provide security to the different users of the system.

2.1.2. Expected Benefits:

- Provide quick up to date information regarding customer details, vendor details, inventory details, etc.
- Transaction time reduction.

2.2 Proposed System

The proposed system Premium Customer Information Management tries to provide a user friendly, GUI based client/server application that will automate the different activities involved in the day to day activities of the QSC.

2.2.1. Selection of Software:

Operating Systems	: Windows NT / 98/95
Database	: Ms SQL Server for database on Server.
GUI Tools	: VISUAL BASIC.NET for Code, Components And Forms. Crystal Reports 6.0 for Reports.

2.2.2. Selection of Hardware:

Server	:	Pentium II/III, 128MB RAM, 20 GB Hard Disk
Clients	:	Pentium II, 128 MB RAM, 10 GB Hard Disk.

2.3 ABOUT VISUAL BASIC .NET

Existing System

Currently there are many site statistics calculator applications are available. Most of these applications work at server rendering the the performance of the server. Few advanced applications optimized their techniques to get the site statistics without sacrificing the server performance. Even some applications gone to the extent of giving individual reports.

The high end advanced applications are not cost effective making them not feasible option to implement at middle and lower level segment. As said earlier all these has to be executed at server side automatically or manually.The major kissing feature of most of the statistics applications are analysis part, though some provide limited analysis results.

Most of these softwares are either created by CGI or PERL or some other server side programming languages leaving them behind the latest upcoming technologies like Visual Studio.Net or forthcoming sun one.

Proposed System

Keeping in view of all the issues specified in the above section, the proposed solution plan provides comprehensive solution to the above addressed issues. This proposed solution namely "Web Log Parser", utilizes the IIS log functionality like most of the competitive products for generating statistical results. Unlike others, this is not Server dependent. The application run from the client systems memory and fetches the information from the IIS logfile at Server or any other specified location at specific intervals.

The retrieval information is parsed using client system resources and stores the information. The main advantages of this new system is it can provide detailed analysis of the logfile. The analysis results can be outputted as grid value blocks or a dynamic Pie\Bar\Line Chart,, giving analytical view of the retrieved data.

From COM to .NET

Visual basic and Visual C++ have separate run times, each with its own distinct behaviors. C++ revolutionized software development by making object_ oriented programming widely used. However C++ objects could only be used by C++ code,which did not benefit most programmers; the majority do not used C++ as a result of the steep learning curve required to
understand it.inorder to solve the problem of inter language communication ,
Microsoft developed COM .

COM is really a contract ,a set of laws that determine how to build a COM
Component.If your component follows the COM rules,it can work with other COM components no matter which language they are written in. When you added a new tool to the Visual Basic 6.0 tool box,that .ocx file was probablywritten in C++, but you did not need to know or care.You just
wanted the functionality,the control provided.

COM deals primarily with interfaces.Not graphical user interfaces,but applicationprogrammin interfaces.In fact ,a COM object is primarily composed of interfaces.If a component adhered to the COM blueprint for how to lay itself out in memory and provided a set of standard interfaces,other COM-compliant software could reuse the component.

COM acts as binary object-interoperability standard.A COM software component is a piece of reusable software in binary form – a self-contained block of functionality such as a grid control ot a text box.. COM components register themselves in the windows registry,and any program that knows how to find them can use them.While this approach has,for the most part worked well ,inadvertent version or interfaces changes by an unwary programmer have caused occasional problems. And when you wanted to spawn a process on a remote machine across the Internet,things rapidly became tricky.

The .NET World

The raison d'etre' of .NET is to provide users with access to their information, files ,or programs anywhere, anytime ,and on any plotform or device.User should not have to know where the information is located or the details about how to retrieve it .For example,over the next several years Microsoft and other companies will phase out delivering software on CDs .
Instead they will deliver the functionality that users get today from software installed on their desktops via Web Services delivered over the Internet . Consumers of those services will no longer buy software, install it on machine , and then maintain it .Instead they will license the functionality as an on demand service . Learning Visual Basic .NET not only puts you on the forefront of exciting and revolutionary vision ,but it also permits you to be more product you with today's applications .Bill Gates summed up things nicely when he told a group of developers recently, " Today ,we have a world of applications and Web sites. In the .NET world , everything that was an application becomes a Web Service." Web Services means the sum is greater than the parts .

A .NET Example

Let's a you're charged with deveping a program that moves founds across different currencies for a financial institution . Designing this program requires that currency conversion rates be calculated in real time, which allows a customer to send a doller-denominated wire transfer and have it converted to deutschemarks(DM) for payment to your branch in Berlin. The customer of your service would have to know ,in real time how many dollars are required to pay the bill of DM 10,000 the moment the wire was sent .You could try to figure out how to create a real-time currency conversion program , but where would you start? How do get the rates? What is the risk of getting it working? How timely is the information ? And on and on.

Programmers might spend months or more just writing the specification and the code for a program like this.then there is testing and debugging , which will take another few months .

Why You Need To Learn Visual Basic . NET

While classic Visual Basic is a powerful and relatively simple programming language, it has reached a " glass ceiling " in addressing the requirements of current technology . For example , classic Visual Basic provides no direct access to underlying APIs, nor does it provide inheritance-the ability to incorporate functionality from another class. Classic Visual Basic programs are also difficult to fine-tune because they are far removed from the underlying machanics of what's going on .

In addition , classic Visual Basic works fine for developing software for the windows platform because Windows 95 , Windows 98 ,Windows Me, Windows NT 4.0, and Windows 2000 all rely on the Intel x86 architecture . but today's developers simply need more horse power, for more horses .

Visual Basic .NET is an entirely new way of developing software . it will let Visual Basic programmers address the advaces in hardware, communications technology, miniaturization , and the Internet , which are all converging at a breakneck pace. I for one don't want to go through again the sort of the development issues encountered when PCs moved from 16-bit to 32-bit architecture and two separate executables had to be tracked and managed depending on the platform a customer was using. In the next year or so,this management nuisance will seem like good times by comparison to the array of different platforms we will have to program for. The new 64-bit Windows is right around the corner ,and more and more companies are relying on the Internet to distribute information and services . these companies want to distribute information to cell phones and wireless hand-held devices. Newer smart devices and appliances are also being developed daily. These wireless devices , as well as Windows CE, which is used in pocket PCs , all run on a non-Intel x 86 CPU and let's not forget about the palm OS based on the Dragonball chip . the obvious problem how can a Visual Basic developer build software and distribute it to all of these disparate devices. With classic Visual Basic , that's inpossible.

Instead of having to learn new techniques to optimize programs for specific hardware and operating systems, use separate architecture-specific compilers for each new(or yet devised)platform, and then track each and every version,developers can use .NET . At the center of the new approach is the common language runtime (CLR). The CLR(which I will describe in more detail later in this chapter)
Provides many benefits to Visual Basic programmers,not the least of which is the means to program in any .NET language- Visual Basic , C++,C# ,or one of 17 others that target the CLR. Components written in

Visual Basic . NET can be called and used by those written in COBOL. NET, for example.

When you program with Visual Basic . NET most applications can be deployed with zero-impact installations ; in other words , installing an application is guaranteed not to affect applications already installed on a system. One of the immediate benefits you will reap is the ability to simply use XCOPY to install your entire application . you no longer have to write complicated and error-prone registry entries. You don't have to use regedit32 to register components. And, one of my personal favourites, you don't have to contend anymore with conflicting o out-of-date DLLs.how many times have you installed a new program only tofind that programs that were working now don't? often this error was caused by an older or incompatible version of a DLL being installed over the current file by the new program. This problem alone caused countless hours of sleuthing, at least in my experience. The Microsoft .NET framework puts an end to DLL Hell once and for all.

In most cases you simply install your visual basic .NET application and all the files associated with it in a directory, and you're done. At first blush, this approach might sound unbeatable, but there is more. Consider when you have to uninstall a classic visual basic program. The results are often unpredictable. Many times the client machine is left with orphaned files and registry entries. With visual basic.NET you simply delete the contents of a directory (or directory tree) and the application is completely removed from the machine. Help desks every where will love this. Where do we start? The reason this chapter is titled "visual basic.NET from the ground up" is that we really do start at the bottom. Conceptually, the visual basic.NET compiler sits atop the .NET framework. The visual basic.NET compiler simply exposes various parts of the .NET framework that are specific to the visual basic language. The visual basic compilier enforces syntax, but all of the real action occurs at the level of the .NET framework.

At the heart of the .NET framework is the common language runtime. The CLR manages execution of .NET code and provides services that make the development process easier. Compliers and tools make the runtime's functionality available. Code that you write that targets the runtime is known as managed code. The CLR manages the code for, among other operations, cross-language integration and exception handling, starting and killing threads, security, versioning and deployment support.

The CLR makes it easy for visual basic .NET developers to deign and build applications whose objects can interact with objects written in other languages. This interaction is possible because the various language compilers and development tools that target the CLR use a common data type system defined by the runtime. Visual basic.NET includes new data types, and older visual basic 6 types, such as the variant, are no longer supported. These changes were to accommodate the CLR specifications. (I'll start discussing the new data types and implications for developers in chapters 2 and 3.) how you experience the runtime depends on which language compiler you use-each one exposes a slightly different subset of runtime functionality. But what the compiler dose expose is identical across languages. This brings to mind one word-interoperability.

Let's take a look at all the moving parts of the .NET framework. As shown in figure 1-1, at the top level is the visual basic compiler (as well as compilers for many other languages). Below the compiler is the common language specification (CLS). This specification is a set of rules that govern the minimum language features that must be supported to ensure that a language will interoperate with other CLS, the code is guaranteed to work with the CLR. In this way, when third-party comilers target the .NET framework, as ong as they conform to the CLS, the ode is guaranteed to run. As you can see from the illustration, vusual basicis now a peer of C++,C#, and any other language targeting .NET. visual basic .NET now has the same variable types, arrays, user-defined types, clases, graphical form , visual controls,and interfaces as these other languages. This common structure makes calling a class in one .NET language from another .NET language effortless.

Figure 1-1 also indicates that you can use the visual basic studio.NET integrated development environment (IDE) to program with visual basic an target the .NET platform. Because the new IDE is similar to the visual basic 6 IDE (although more streamlined), visual basic programmers will immediately feel at home.

An overview of the .NET framework.
Web services

Web services provides a web-enabled user interface with tools that include various hypertext markup language (HTML) controls and web controls. Web services also handle various web protocols, security, when code is targeted for .NET, it is called managed code, which means that the code automatically runs under a "contract of cooperation" with the CLR. Managed code supplies the information necessary for the runtime to provide services such as memory management, cross-language integration, code access security, and the automatic lifetime control of all of our objects.

Don't worry if trying to digest all this information seems like trying to drink water from a fire hose. As you start building your first visual basic .NET programs, you'll rapidly see how the pieces fit together.
Abstraction layers
Where do we start to access functionality from visual basic .NET source code?
What you might notice first when looking over this architecture is that all related logical functionality is grouped in the .NET frameworks components for easy management. You can access this functionality in your programs by referring to what's called a namespace. Namespaces are a hierarchical naming scheme for grouping types into logical categories of related functionality in .NET. for example, when you create your first form, the IDE will add imports system. Windows. Forms to the code. The new imports directive is similar to adding references in visual basic 6.0 importing a namespace is all you need to do to inherit the functionality from the user interface block in order to build and draw forms. I suspect that you are starting to see the elegance and power of this approach.

What I hope is also starting to become clear is that .NET provides a host of services that are there for the taking, finally making the holy grail of code reuse a reality. These services, along with other benefits, are designed to increase our productivity over the course of development- from design and coding to deployment and security. You can continue to develop stand-alone applications for Pc's with visual basic .NET, but you can also handle any task on any platform that the job demands or might demand in the future.

Abstraction layers

The idea of an abstraction layer is not new. The concept is an extension of the hardware abstraction layer (HAL) in the windows operating system. Before windows 3.0, if you wrote a program for a pc, you had to know the answer to all sorts of questions. For example, how do I print a document? Because each printer has its own driver and printer control language, you had to write software that accommodated and spoke to all sort of printers. What monitor was used? Did the pc have expanded or extended memory? Sheet! This code was incredibly difficult to write. Then the windows operating system came on the scene, it became the great equalizer. All you had to do was program to windows, and it took care of the messy details. You could just say "print" and windows figured out how to do it. Windows handled monitor resolution, memory management, and all the other nagging details that had to be handled by programmers in the DOS world. With .NET, the CLR is also the great equalizer but on a grander scale. Instead of just handling individual printers and monitors, it handles the differences in the underlying hardware architecture. This capability is nothing short of amazing and will provide and incredible boost in productivity immediately.

Visual Basic .NET is object oriented

As a Visual Basic programmer , you might be thinking that up till now you did not need to learn object-oriented programming. Classic Visual Basic served your needs nicely.
You could do most anything you needed to with some advanced knowledge of the language. However, many programming luminaries feel that the shift fro the current programming paradigm to .NET will be at least if not more monumental that the shift from DOS to windows. A new, energized object-oriented Visual Basic is the tool that is needed.

If you are new to object-oriented programming ,don't let this worry you.. As I cover the new facets of language ,I'll also explain and illustrate object –oriented techniques.The concepts presented in this chapter(and described in more detail in chapter 2,"object-oriented programming in Visual Basic.NET") will be amplified throughout the book..If concepts aren't completely clear after you finish this chapter,they soon will be.

Remember that with power comes responsibility .Classes and objects were first introduced in Visual Basic 4.0 and enhanced in versions 5.0 and 6.0 .Unfortunately ,studies show that over 50 percent of all Visual Basic programmers chose to disregard them.While we could safely turn

our backs on classes and objects in previous versions of Visual Basic ,we can no longer ignore them with Visual Basic .NET.

As in Java,every thing in Visual Basic .NET is an object .But what exactly does that mean?well,something like an integer object now has its own methods that can format the value the object had or change itself into a string.Object –orientation is part and parcel of Visual Basic .NET,so you didn't want to avoid it any longer.

To be honest with you from the start ,Visual Basic .NET will require a steeper learning curve than classic Visual Basic .I will cover all aspects of Visual Basic .NET programming ,including concepts such as events and classes ,which some programmers find difficult.Each part will be covered in detail .And once you can learn Visual Basic .NET,you can very easily move to C# or even Visual C++.NET.A welcome by –product of learning Visual Basic .NET now is that this knowledge will also expand your understanding and use of other languages,which will make you more marketable.

Unlike programming in previous of Visual Basic ,programming in Visual Basic .NET requires an understanding of not only the framework I just covered ,but the language itself and the infrastructure of how.NET programs are assembled .When teaching Visual Basic 6 to my graduate students,I hold off covering classes and objects until later in the semester.We pace ourselves because there is so much you can do without objects in classic Visual Basic.Because we don't have that luxury in this book,the first few chapters will be front loaded with new concepts such as the .NET framework ,how Visual Basic .NET programs are built,and the fundamentals of object-oriented programming .The remainder of the book will provide programs to hammer home these concepts.

Of course,classic Visual Basic was designed for simplicity ,and while simplicity accounted for Visual Basic's phenomenal success ,it also meant the language evolved along a different path from C++.with Visual Basic ,creating applications with various buttons ,text boxes ,and other graphical gizmos was relatively straightforward .However ,object-oriented capabilities such as inheritance just weren't part of the language.Visual Basic .NET has lost some of this simplicity in favor of more power ,flexibility ,and robustness.Visual Basic has grown up and joined the ranks of other pure object-oriented languages to handle the programming tasks of the twenty-first century.

As classic Visual Basic evolved ,it became more complex.Programmers could build DLLs,ActiveX controls and servers ,and classes.Along with this power came a level of difficulty.Advanced Visual Basic programmers had to know about the Windows API,COM,DCOM,ADO ,classes,objects,and the rest.Likewise ,classic Visual C++ evolved by getting easier.Various graphical tools ,frameworks,wizards ,code generators,and templates were added to assist in handling the lower – level tasks of Windows programming.
As Visual Basic grew in complexity and language power.At that intersection is .NET.Both languages have evolved to a point at which they are now relatively equal complexity .As I said earlier ,.NET is the great equalizer.

The Visual Basic .NET Web Forms and the Windows Form designers permit a developer to easily create standardized interfaces.Bo Base Class Library The base class library(BCL) is underneath the Data and XML block.
This area is the origin for the base class of all.NET programs.Everything in Visual Basic .NET is an object,and all objects originate from a class named System .The BCL also provides collections ,localization ,text objects ,interoperability with non-.NET code and ActiveX controls ,and a variety of other services.

Most classic Visual Basic programmers are familiar with APIs . But C++ programmers use the Microsoft Foundation Classes(MFC),while developers using Java to create Windows applications rely on yet a different framework . The unfied programming classes in the BCL ensure that all languages use this same object-oriented , hierarchical , and extensible set of class libraries.

Common Language Runtime

Holding up the framework is the common language runtime , which I described briefly earlier. The CLR does the heavy lifting for .NET Framework programs. Of course , runtimes are nothing new for programming languages . To run a classic Visual Basic programs the VBRUN run time has to be installed on the client machine. Likewise Visual C++ must have MSVCRT installed , Java must have its won run time installed , and so on. Only the CLR is needed to make .NET code run on any machine. The CLR is a set of standed resources that any .NET program can take advantage of , from any .NET-supported language. All

languages will be more equal in capability than they have ever been before-.NET is the grate equalizer . But not all .NET languages will support all .NET services. You can perform some operations in Visual Basic .NET operations that you can't do in C# ,and vice versa. The capabilities of each languages are dictated by the language's compiler.

The CLR includes support for the BCL,where the architecture for our controls and forms actually live. It is also responsible for managing threads and exceptions (what were errors held in the Err object in classic Visual Basic).Garbage collection-or relasing objects that are no longer in use-is also done by this CLR.(I'll touch on garbage collection again later in this chapter and describe it more detail in chapter 4, "Visual Basic .NET Data Types and Features.") the CLR takes the code generated by Visual Basic . NET compiler(or any other compiler that targets .NET , for that matter) and converts it to the native language of the current platform architecture. Throw this conversion the magic of inter platform execution is achieved . Visual Basic programmers write code in the Visual Basic syntax,and the CLR is responsible for converting it to any platform that can run the CLR. Programmers are abstracted several levels from the hardware and really don't need to know or care what platform their code will be run on. As long as we use proper Visual Basic syntax and the application can be built without the compiler barking at us, . NET takes care of the rest.

As you can see in figures 1-1 and 1-2 , the CLR provides support for everthing above it, as well as the conversion of Visual Basic .NET code to the specific architecture an application is running on . It also atonatically provides type-safe code , which means that the code can do only what we want it to do . you have all heard stories about rogue code that either through sloppiness or maliciousness overran buffers and trashed a machine , killing all other programs on board . By design , this situation can't happen with Visual Basic .NET code.

The common language runtime provides a number of powerful services.

A form was added to a project, all the complexity of construction was hidden. All of the public interfaces were hidden from the programmer. However , they were always there , and the form was still a class under the hood . In Visual Basic .NET , a Windows Forms module contains all of the code to instantiate (create an instance of) itself as well as any controls placed on the form . Programmers are responsible for adding code for handling events. In Visual Basic . NET , you can't even create a form without understanding the concept of a class .

XML is rapidly becoming the lingua of franca of the Web . You have probably read that XML has taken center stage for moving data across the Internet ,so it is central to the .NET strategy . In a simple program we create later in this chapter, you'll see what the format of XML looks like. I don't cover XML basics in this book . you should become familiar with XML to work effectively with Visual Studio . NET .

XML Integration with ADO.NET

Synchronizing a DataSet with an XmlDataDocument

When using previous versions of Activex Data Objects (ADO), the code written when working with relational data was different than the code written to work with hierarchical data. The .NET architecture integrates several classes in XML with several classes in the ADO.NET architecture to unify the two programming models. This gives you an object model to use regardless of the structure of the data.

The XML in the .NET Framework and ADO.NET provide a unified programming model to access data represented as both XML data, text delimited by tags that structure the data, and relation data, tables consisting of rows, columns, and constraints. The XML reads XML data from any data stream into DOM node trees, where data can be accessed programmatically, while ADO.NET provides the means to access and manipulates relational data within a DataSet object. Of the classes that comprise XML in the .NET Framework and ADO.NET, the DataSet represents a relational data source in the ADO.NET , the XmlDocument implements the DOM in XML, and the XmlDataDocument unifies the ADO.NET and XML by representing relational data from a DataSet and synchronizing it with the XML document model.

The XML integration with relational data occurs the XmlDataDocument, a class derived from the XmlDocument. The XmlDataDocument maps XML to relational data in an ADO.NET DataSet.

Advantages Of the XML Format

In some important ways,XML is just another data format. In other ways, XML has several key advantages over other formats that have helped catpult it forward other approaches to storing information.

1. XML allows developers to create their own labeled structures for storing information.
2. XML parsing is well-defined and widely-implemented, making it possible to retrieve information from XML documents in a variety of environments.
3. XML is built on a Unicode foundation,making it easier to create internationalized documents.
4. applications can rely on XML parsers to do some structural validation, as well as data type checking.
5. .XML formats are text-based,making them more readable, easier to document sometimeseasier to debug.
6. Tools are available for XML processing on different platforms, making it simpler to use.
7. XML instead of binary formats to exchange complex information streams.
8. XML documents can use much of the infrastructure already built for HTML, including the HTTP protocol and some browsers.
9. XML isn't appropriate for every situation, however. XML documents tend to be more Verbose than the binary formats they replace.
10. They take up more network bandwidth and storage space, or require more processor time for compression. XML parsing can be slower than optimized binary formats and can require more memory parsing highly. However, careful application design cn avoid some of these problems.

Architetural Overview of XML in the .Net Framework

The design goals for the XML classes in the .NET Framework are:

1. High- productivity.
2. Standards-based.

3. Multilingual support.
4. Extensible.
5. Pluggable architecture.
6. Focused on performance, reliability, and scalability.

Integration with ADO.NET.

The .NET Framework provides an opportunity to design an integrated suite of XML classes and also show innovation in the XML world. The XML class provided are core elements of the .NET Framework. These classes provide an open, standards-compliant, interoperable solution to the challenges that developers face today.

Overview of ADO.NET

ADO.NET provides consistent access to data sources such as Microsoft SQL Sever, as well as data sources exposed via OLEDB and XML. Data Sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, manipulate, and update data.

ADO.NET cleanly factors data access from data manipulation into discrete components that can be used separately or in tandem. ADO.NET includes . NET data providers for connecting to a database, executing commands, and retrieving results.

Those results are either proessed directly, or lced in an ADO.NET **data set**
Object in order to be exposed t the user in an ad-hoc manner combined with data from multiple sources or remoted between tiers.
The ADO.NET **data set** object can also be used independently of a .NET data provider to manage data local to the application or sourced from XML.

The ADO.NET classes are found in system. data.dll, and are integrated with the XML classes found in system. Xml. dll.when compiling code that uses the system. Data namespace, reference both system. Data.dll and system. Xml.dll compiler.

For an example of compiling an ADO.NET application using a command line ADO.NET provides functionality to developers writing managed

code similar to the functionality provided to native COM developers by ADO.

Discusses the motivation and design goals behind creation ADO.NET.
Provides an overview of the architecture and components of ADO.NET.
Provides an overview of the design of the .NET data provider and of the .NET data providers that are included with ADO.NET.
Provides an overview of the **data set** design and components.

Provides information about hw to use the common interfaces supplied by ADO.NET to write a single set of code that will work regardless of the . NET data provider.

Provides an example of an ADO.NET application that retrieves data from a database and returns it to the console.

Describes the ADO.NET architecture and components and how t use them to access existing data sources as well as to manage application data.

Connecting to SQL Server Using ADO.NET [Visual Basic]

the SQL server .NET data provider provides connectivity to Microsoft SQL server version 7.0 or later using the **SQL connection** object.
The SQL server.NET data provider supports a connection string format that is similar to the OLEDB(ADO) connection string format. For valid string format names and values, see the sql connection. Connection string property.

The following code example demonstrates how to create and open a connection to a SQL server(version 7.0 or later) data base.

[Visual Basic]

```
dim nwind conn as sql connection = new sql connection (“data source=localhost; integrated security=sspi ”&_”initial catalog=north wind”)
nwindconn.open ()
```

[c#]

```
sql connection nwindconn=new sql connection (“data source=local host;integrated security=sspi;”+”initial catalog=northwind”);
nwindconn.open ()
```

closing the connection

you must always close the connection when you are finished using it. This can be done using either the **Close** or **Dispose** methods of the **Connection** object. Connections are not implicitly released when the **Connection** object falls out of scope or is reclaimed by garbage collection.

Design Goals for ADO.NET

As application development has evolved, new applications have become loosely coupled based on the web application model. More and more of today’s applications use XML to encode data to be passed over network connections. Web applications use HTTP as the fabric for communication between tiers, and therefore must placidly handle maintaining state between requests. This new model is very different from the connected, tightly coupled style of programming that characterized the client/server era, where a connected was held open for the duration of the program’s lifetime and no special handling of state was required.

In designing tools and technologies to meet the needs of today's developer, Microsoft recognized that an entirely new programming model for data access was needed, one that is built upon the .NET framework. Building on the .NET framework ensures that the dta access technology would be uniform components would share a common type system, design patterns, and naming conventions.

ADO.NET was designed to meet the needs of this new programming model:
Disconnected data architecture, tight integration with XML, common data representation with the ability to combine data from multiple and varied data sources, and optimized facilities for interaction with a database, all native to the .NET framework.
In creating ADO.NET, Microsoft embraced the following design goals.

Leverage Current ADO Knowledge

The design for ADO.NET addresses many of the requirements of today's application development model. At the same time, the programming model stays as similar as possible to ADO, so current ADO developers do not have to start from the beginning in learning a brand new data access technology. ADO.NET is an intrinsic part of the .NET framework without seeming completely foreign to ADO programmer.

ADO.NET coexist with ADO. While most new .NET-based applications will be written using ADO.NET, ADO remains available to the .NET programmer through .NET COM interoperability services.

Support the N-Tier programming model

ADO.NET provides first-class support for the disconnected, n-tier programming environment for which many new applications are written. The concept of working with a disconnected set of data has become a focal point in the programming model. The ADO.NET solution for n-tier programming is the **data set.**

Integrate XML Support

XML and dta access are intimately tied-XML is all about encoding data, and data access is increasingly becoming all about XML. The .NET

framework does not just support web standards-it is built entirely on top of them.

XML support is built into ADO.NET at a very fundamental level. The XML classes in the .NET framework and ADO.NET are part of the same architecture-they integrate at many different levels. You no longer have to choose between the data access set of services and their XML counterparts; the ability to cross over from one to the other is inherent in the design of both.

ADO.NET Architecture [Visual Basic]

Data processing has traditionally relied primarily on a connection-based, two-tier model. As data processing increasingly uses multi-tier architectures, programmers are switching to a disconnected approach to provide better scalability for their applications.

XML and ADO.NET

ADO.NET leverages the power of XML to provide disconnected access to data. ADO.NET was designed hand-in-hand with the XML classes in the .NET framework-both are components of a single architecture.

ADO.NET and the XML classes in the .NET framework converge in the **data set** object. The **data set** can be populated with data from an XML source, whether it is a file or an XML stream. The **data set** can be written as World Wide Web consortium(W3C)compliant XML, including its schema as XML schema definition language(XSD)schema, regardless of the source of the data in the **data set.** Because the native serialization format of the **data set** is XML, it is an excellent medium for moving data between tiers making the **data set** an optimal choice for remoting data and schema context to and from an XML Web service.

The **data set** can also be synchronized with an **Xml data document** to provide relational and hierarchical access to data in real time.

ADO.NET Components

The ADO.NET components have been designed to factor data access from data manipulation. There are two central components of ADO.NET that accomplish this: the **dataset,** and the .NET data provider, which is a set of components including the **Connection, Command, Data Reader,** and **Data Adapter** objects.

The ADO.NET is the core component of the disconnected architecture of ADO.NET. The **data set** is explicitly designed for data access independent of any data source. As a result it can be used with multiple and differing data sources, used with XML data, or used to manage data local to the application. The **data set** contains a collection of one or more **data table** objects made up of rows and columns of data, as well as primary key, foreign key, constraint, and relation information about the data in the **data table** objects.

The other core element of the ADO.NET architecture is the whose components are explicitly designed for data manipulation and fast, forward-only, read-only access to data. the **connection** object provides connectivity to data source.

The **command** object enables acss to database commands to return data, modify data, run stored procedures, and send or retrieve parameter information. The **data reader** provides a high-performance stream of data from the data source. Finally, the **data adapter** uses **command** objects to execute SQL commands at the data source to both load the **data set** with data, and reconcile changes made to the data in the **data set** back to the data source.

You can write .NET data providers for any data source. The .NET framework ships with two .NET data providers: the SQL server .NET data provider and the OLEDB.NET data provider.

ADO.NET architecture

Remoting or marshaling data between tiers and clients

The design of the **dataset** enables you to easily transport data to clients over the web using XML web services, as well as allowing you to marshal data between .NET components using .NET remoting services. You can also remote a strongly typed **dataset** in this fashion.

ADO.NET Platform Requirements

The Microsoft .NET framework SDK (including ADO.NET) is supported on microsoft

Remoting or Marshaling Data between Tiers and Clients

The design of the DataSet enables you to easily transport data to clients over

the Web using XML Web services , as well as allowing you to marshal data

between .NET components using .NET Remoting services.You can also

remote a strongly typed DataSet in this fashion.

ADO.NET Platform Requirements

The Microsoft.NET Framework SDK (including ADO.NET) is supported on

Microsoft Windows 2000.Microsoft Windows NT 4 with Service

Pack6a.Microsoft Windows Millenium Edition.Microsoft Windows 98 and

Microsoft Windows SE . Use of the SQL Server .NET Data provider or

OLEDB.NET Data Provider require the installation of Microsoft Data

Access components version 2.6 or later.

The following code example shows how to include the System.Data

namespace in your applications,in order to use Ado.Net.

[Visual Basic]

Imports System.Data

[c#]

using System.Data;

The ADO.NET classes are found in System.Data.dll,and are integrated with

the XML classes are found in System..dll.when compiling code that uses the

System.Data namespace,reference both system.Data.dll and

System.Xml.dll.for an example of compiling an ADO.net application using a

command line compiler.

.NET Data Providers [Visual Basic]

A .NET data provider is used for connecting to a database, executing

commands, and retreiving results. Those results are either processed directly,

or placed in an ADO.NET Data set in order to be exposed to the user in an

ad-hoc manner, combined with datafrom multiple sources , or remoted

between tiers. The .NET data provider is designed to be lightweight, creating

a minimal layer between the datasource and your code, increasing

performance without sacrificing functionality.

The following table outlines the four core objects that make up a .NET data

provider.

Object	Description
Connection	Establishes a connection to a specific datasource.

Command	Executes a command against a data source. Exposes Parameters and can execute with in the scope of a Transaction from a Connection
DataReader	Reads a forward-only, read-only stream of data from a data source.

DataAdapter Populates a DataSet and resolves updates with the data source.

The .NET Framework includes the SQL Server .NET Data Provider (for Microsoft SQL Server Version 7.0 or later), and the OLEDB.NET Data Provider.

An Open Database Connectivity (ODBC).NET Data Provider is available as a separate download at http: //msdn.microsoft.com/downloads. The download includes documentation on the classes that make up the ODBC.NET Data Provider. However, the implementation has the same architecture as bothe the SQL Server.NET Data Provider and the OLEDB.NET Data Provider as well.

The SQL Server.NET Data Provider

The SQL Server.NET Data Provider uses its own protocol to communicate with SQL Server. It is lightweight and performs well because it is optimized to access a SQL Server directly without adding an OLEDB or Open

Database Connectivity (ODBC) layer. The following illustration contrasts the SQL Server.NET Data provider with the OLEDB.NET Data Provider.

The OLEDB.NET Data Provider communicates to an OLEDB data source

through both the OLEDB service component, which provides connection pooling and transaction services and the OLEDB Provider for the data source.

Comparison of the SQL Server.NET Data Provider and the OLE DB .NET Data Provider

To use the SQL Server .NET Data Provider, you must have access to

Microsoft SQL Server 7.0 or later. SQL Server .NET Data Provider classes

are located in the System.Data.SqlClient namespace. For earlier versions of

Microsoft SQL Server, use the OLE DB.NET Data Provider with the SQL

Server OLEDB Provider(SQLOLEDB).

The following code example shows how to include the

System.Data.SqlClient namespace in your applications.

```
[Visual Basic]
Imports System.Data.SqlClient
[C#]
using System.Data.SqlClient;
```

The SQL Server.NET Data Provider requires the installation of Microsoft Data Access Components(MDAC) version 2.6 or later.

The OLE DB.NET Data Provider

The OLE DB.NET Data Provider uses native OLE DB through COM

interop to enable data access. The OLE DB.NET Data Provider supports both manual and automatic transactions. For automatic transactions, the OLE DB.NET Data Provider automatically enlists in a transaction and obtains transaction details from Windows 2000 Component Services.

To use the OLEDB.NET Data Provider , you must use an OLE DB Provider

that supports the OLEDB interfaces.

The following table shows the providers have been tested with ADO.NET.

Driver	Provider
SQLOLEDB	Microsoft OLE DB Provider for SQL Server.
MSDAORA	Microsoft OLE DB Provider for Oracle.
Microsoft.Jet.OLEDB.4.0	OLE DB Provider for Microsoft Jet

The OLEDB.NET Data Provider does not support OLEDB version 2.5 Interfaces. OLEDB Providers that require support for OLEDB 2.5 interfaces

function properly with the OLE DB.NET Data Provider. This includes the

Microsoft OLE DB Provider for Exchange and the Microsoft OLE DB

Provider for Internet Publishing.

The OLE DB.NET Data Provider does not work with the OLE DB Provider

for ODBC (MSDASQL).To access an ODBC data source using ADO.NET,

use the ODBC.NET Data Provider, which is available for download at http : // msdn.microsoft.com /downloads

OLE DB.NET Data Provider classes are located in the System.Data.Oledb

namespace.The following code example shows how to include the

System.Data.Oledb namespace in your applications.
[Visual Basic]
Imports System.Data.Oledb
[C#]
using System.Data.Oledb;
The OLE DB.NET Data Provider requires the installation of MDAC 2.6 or
Later.

Choosing a .NET Data Provider

Depending on the design and data source for your application, your choice

of .NET data provider can improve the performance,capability, and integrity

of your application. The following table discusses the advantages and

limitations of each .NET Data Provider.

Provider	Notes
	Recommended for middle-tier applications using Microsoft SQL Server7.0 or later.

	Recommended for single-tier applications using Microsoft Data Engine (MSDE) or Microsoft SQL Server 7.0 or later.
SQLServer .NET Data with Provider Provider.For	Recommended over use of the OLE DB Provider for SQL Server(SQLOLEDB) the OLE DB.NET Data Microsoft SQL Server version 6.5 and earlier, you must use the OLE DB Provider for SQL server with the OLEDB.NET Data Provider. Recommended for middle tier applications using Microsoft SQL server 6.5 or earlier or any OLEDB provider that supports the OLE DB interfaces listed in (OLEDB 2.5 interfaces are not required). For Microsoft SQL Server7.0 or later, the SQL Server.NET DataProvider is recommended.
OLEDB.NET Data Provider	Recommended for single-tier applications using Microsoft Access databases. Use of a Microsoft Access Database for a middle-tier application is not recommended. Support for the OLE DB Provider for ODBC(MSDASQL) is disabled. For access to Open Database Connectivity (ODBC) data sources, an ODBC.NET Data Provider is available as a separate download at http:// msdn.microsoft.com/downloads.

2.4 About Microsoft SQL Server 7.0

Microsoft SQL Server is a Structured Query Language (SQL) based, client/server relational database. Each of these terms describes a fundamental part of the architecture of SQL Server.

Database

A database is similar to a data file in that it is a storage place for data. Like a data file, a database does not present information directly to a user; the user runs an application that accesses data from the database and presents it to the user in an understandable format.
A database typically has two components: the files holding the physical database and the database management system (DBMS) software that applications use to access data. The DBMS is responsible for enforcing the database structure, including:

- Maintaining the relationships between data in the database.
- Ensuring that data is stored correctly and that the rules defining data relationships are not violated.
- Recovering all data to a point of known consistency in case of system failures.

Relational Database

There are different ways to organize data in a database but relational databases are one of the most effective. Relational database systems are an application of mathematical set theory to the problem of effectively organizing data. In a relational database, data is collected into tables (called relations in relational theory).
When organizing data into tables, you can usually find many different ways to define tables. Relational database theory defines a process, normalization, which ensures that the set of tables you define will organize your data effectively.

Client/Server

In a client/server system, the server is a relatively large computer in a central location that manages a resource used by many people. When individuals need to use the resource, they connect over the network from their computers, or clients, to the server.
Examples of servers are: In client/server database architecture, the database files and DBMS software reside on a server. A communications component is provided so applications can run on separate clients and communicate to the database server over a network. The SQL Server communication component also allows communication between an application running on the server and SQL Server.

Server applications are usually capable of working with several clients at the same time. SQL Server can work with thousands of client applications simultaneously. The server has features to prevent the logical problems that occur if a user tries to read or modify data currently being used by others.

While SQL Server is designed to work as a server in a client/server network, it is also capable of working as a stand-alone database directly on the client. The scalability and ease-of-use features of SQL Server allow it to work efficiently on a client without consuming too many resources.

Structured Query Language (SQL)

To work with data in a database, you must use a set of commands and statements (language) defined by the DBMS software. There are several different languages that can be used with relational databases; the most common is SQL. Both the American National Standards Institute (ANSI) and the International Standards Organization (ISO) have defined standards for SQL. Most modern DBMS products support the Entry Level of SQL-92, the latest SQL standard (published in 1992).

SQL Server Features

Microsoft SQL Server supports a set of features that result in the following benefits:

Ease of installation, deployment, and use

SQL Server includes a set of administrative and development tools that improve your ability to install, deploy, manage, and use SQL Server across several sites.

Scalability

The same database engine can be used across platforms ranging from laptop computers running Microsoft Windows® 95/98 to large, multiprocessor servers running Microsoft Windows NT®, Enterprise Edition.

Data warehousing

SQL Server includes tools for extracting and analyzing summary data for online analytical processing (OLAP). SQL Server also includes tools for visually designing databases and analyzing data using English-based questions.

System integration with other server software

SQL Server integrates with e-mail, the Internet, and Windows.

Databases

A database in Microsoft SQL Server consists of a collection of tables that contain data, and other objects, such as views, indexes, stored procedures, and triggers, defined to support activities performed with the data. The data stored in a database is usually related to a particular subject or process, such as inventory information for a manufacturing warehouse.

SQL Server can support many databases, and each database can store either interrelated data or data unrelated to that in the other databases. For example, a server can have one database that stores personnel data and another that stores product-related data. Alternatively, one database can store current customer order data, and another; related database can store historical customer orders that are used for yearly reporting. Before you create a database, it is important to understand the parts of a database and how to design these parts to ensure that the database performs well after it is implemented.

System Design

Design of software involves conceiving planning out and specifying the externally observable characteristics of the software product. We have data design, architectural design and user interface design in the design process. These are explained in the following section. The goal of design process is to provide a blue print for implementation, testing and maintenance activities.

DATA DESIGN:

The primary activity during data design is to select logical representations of data objects identified during requirement analysis and software analysis. A data dictionary explicitly represents the relationships among data objects and the constrains on the elements of the data structure. A data dictionary should be established and used to define both data and program design.

FESIBILITY STUDY:

Feasibility study is conducted once the problem is clearly understood. Feasibility study is a high level capsule version of the entire system analysis and design process. The objective is to determine quickly at a minimum expense how to solve a problem. The purpose of feasibility is not to solve the problem but to determine if the problem is worth solving. The system has been tested for feasibility in the following points.

a) Technical Feasibility
b) Economical Feasibility
c) Operational Feasibility

a. Technical Feasibility: -

The project entitles "Premium Customer Information Management" is technically feasible because the project was developed in High end technologies like VISUAL BASIC.NET as front-end and Sqlserver 2000 as backend which runs on all the major operating systems like windows 9x, winnt, win 2x, Mac etc.,

b. Economical Feasibility: -

The computerized system will help in automate the selection leading the profits and details of the organization. With this software, the machine and manpower utilization are expected to go up by 80-90% approximately. The costs incurred of not creating the system are set to be great, because precious time can be wanted by manually.

c. Operational Feasibility: -

In this project, the management will know the details of each project where he may be presented and the data will be maintained as decentralized and if any inquires for that particular contract can be known as per their requirements and necessaries.

IMPLEMENTATION:

Implementation is the stage where the theoretical design is turned into a working system. The most crucial stage in achieving a new successful system and in giving confidence on the new system for the users that it will work efficiently and effectively.

The system can be implemented only after thorough testing is done and if it is found to work according to the specification.

It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the change over and an evaluation of change over methods a part from planning. Two major tasks of preparing the implementation are education and training of the users and testing of the system.

The more complex the system being implemented, the more involved will be the systems, analysis and design effort required just for implementation.

The implementation phase comprises of several activities. The required hardware software acquisition is carried out. The system may require some software to be developed. For this, programs are written and tested. The user then changes over to his new fully tested system and the old system is discontinued.

TESTING:

The testing phase is an important part of software development. It is the process of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied.

Software testing is carried out in three steps:

The first includes unit testing, where in each module is tested to provide its correctness, validity and also determine any missing operations and to verify whether the objectives have been met. Errors are noted down and corrected immediately. Unit testing is the important and major part of the project. So errors are rectified easily in particular module and program clarity is increased. In this project entire system is divided into several modules and is developed individually. So unit testing is conducted to individual modules.

The second step includes Integration testing. It need not be the case, the software whose modules when run individually and showing perfect results, will also show perfect results when run as a whole. The individual modules are clipped under this major module and tested again and verified the results. This is due to poor interacting, which may results in data being lost across an interface. A module can have inadvertent, adverse effect on any other or on the global data structures, causing serious problems.

The final step involves validation and testing which determines which the software testing which determines which the software functions as the user expected. Here also some modifications were. In the completion of the project it is satisfied fully by the end user.

MAINTENANCE AND ENHANCEMENT:

As the number of computer based systems, grieve libraries of computer software began to expand. In house developed projects produced tones of thousand soft program source statements.

Software products purchased from the outside added hundreds of thousands of new statements. A dark cloud appeared on the horizon. All of these programs, all of those source statements-had to be corrected when false were detected, modified as user requirements changed, or adapted to new hardware that was purchased. These activities were collectively called Software Maintenance.

The maintenance phase focuses on change that is associated with error correction, adaptations required as the software's environment evolves, and changes due to enhancements brought about by changing customer requirements. Four types of changes are encountered during the maintenance phase.

Correction

Adaptation

Enhancement

Prevention

Correction:

Even with the best quality assurance activities is lightly that the customer will uncover defects in the software. Corrective maintenance changes the software to correct defects.

Maintenance is a set of software engineering activities that occur after software has been delivered to the customer and put into operation. Software configuration management is a set of tracking and control activities that began when a software project begins and terminates only when the software is taken out of the operation.

We may define maintenance by describing four activities that are undertaken after a program is released for use:

Corrective Maintenance

Adaptive Maintenance

Perfective Maintenance or Enhancement

Preventive Maintenance or Reengineering

Only about 20 percent of all maintenance work is spent "fixing mistakes". The remaining 80 percent are spent adapting existing systems to changes in their external environment, making enhancements requested by users, and reengineering an application for use.

Adaptation:

Over time, the original environment (E.g., CPU, operating system, business rules, external product characteristics) for which the software was developed is likely to change. Adaptive maintenance results in modification to the software to accommodate change to its external environment.

Enhancement:

As software is used, the customer/user will recognize additional functions that will provide benefit. Perceptive maintenance extends the software beyond its original function requirements.

Prevention:

Computer software deteriorates due to change, and because of this, preventive maintenance, often called software reengineering, and must be conducted to enable the software to serve the needs of its end users. In essence, preventive maintenance makes changes to computer programs so that they can

be more easily corrected, adapted, and enhanced. Software configuration management (SCM) is an umbrella activity that is applied throughout the software process. SCM activities are developed to
Identity change
Control chug.
Ensure that change is being properly implemented
Report change to others that may have an interest.

3.1 Design Methodology

The two basic modern design strategies employed in software design are

1. Top Down Design
2. Bottom Up Design

Top down Design is basically a decomposition process, which focuses on the flow of control. At later stages it concerns itself with the code production. The first step is to study the overall aspects of the tasks at hand and to break it into a number of independent modules. The second step is to break each one of these modules further into independent sub-modules. The process is repeated one to obtain modules, which are small enough to group mentally and to code in a straightforward manner. One important feature is that at each level the details of his design at the lower level are hidden. Only the necessary data and control that must be called back and forth over the interface are defined.

In a bottom-up design one first identifies and investigates parts of design that are most difficult and necessary designed decision are made the remainder of the design is tailored to fit around the design already chosen for crucial part. It vaguely represents a synthesis process explained in previous section.

One strong point of the top-down method is that it postpones details of the decision until the last stage of the decision. It allows making small design changes when the design is half way through. There is danger that the specifications will be incompatible and this will not be discovered until late in the design process. By contrast the bottom-up strategy first focuses on the crucial part so that feasibility of the design is tested at early stage.

In mixing top-down and bottom-up design it often appears that we start in the middle of the problem and work our way both up and down there. In a complex problem; it is often difficult to decide how to modularize the various procedures in such cases one might consider a list of system inputs and decide what functions are necessary to process these inputs. This is called back to front design. Similarly one can start with the required outputs and work backwards evolving so called front-back design. We have applied both the top down and bottom up approach in our design approach.

3.2 Database Design

Databases are normally implemented by using package called a Data Base Management System (DBMS). Each particular DBMS has somewhat unique characteristics, and as such, general techniques for the design of databases are limited. One of the most useful methods of analyzing the data required by the system for the data dictionary has developed from research into relational databases, particularly the work of E.F.Codd. This method of analyzing data is called "Normalization". Normalized data are converted into normalized data by three stages. Each stage has a procedure to follow.

Normalization:

The first stage in normalization is to reduce the data to its first normal form, by removing repeating items and showing them as separate records but including in them the key fields of the original record.

The next stage of reduction to the second normal form is to check that the record which one is in first normal form, all the items in each record are entirely dependent on the key of the record. If a data item is not dependent on the key of the record, but on the other data item, then it is removed with its key to form another record. This is done until each record contains data items, which are entirely dependent on the key of their record.

The final stage of the analysis, the reduction of third normal form involves examining each record, which one is in second normal form to see whether any items are mutually dependent. If there are any item there are removed to a separate record leaving one of the items behind in the original record and using that as the key in the newly created record.
Here the tables are normalized up to the third normal form.

3.3 Code Design

The Code is designed with the following characteristics in mind.

1. Uniqueness: The code structure must ensure that only one value of the code with a single meaning are correctly applied to a given entity or attribute.

2.Expandibility: The code structure are designed for in a way that it must allow for growth of it's set of entities or attributes, thus providing sufficient space for the entry of new items with in each classification.

3. Consiseness: The code requires the fewest possible number of positions to include and define each item.

4. Uniform size and format: Uniform size and format is highly desirable in mechanized data processing systems. The addition of prefixes and suffixes to the root code should not be allowed especially as it is incompatible with the uniqueness requirement.

5. Simplicity: The codes are designed in a simple manner to understand and simple to apply.

6. Versatility: The code allows modifying easily to reflect necessary changes in conditions, characteristics and relationships of the encoded entities. Such changes must result in a corresponding change in the code or coding structure.

7. Sortability: Reports are most valuable for user efficiency when sorted and presented in a predetermined format or order. Although data must be sorted and collated, the representative code for the date does not need to be in a storable form if it can be correlated with another code that is sortable.

8. Stability: Codes that do not require to be frequently updated also promote use efficiency. Individual code assignments for a given entity should be made with a minimal likelihood of change either in the specific code or in the entire coding structure.

9. Meaningfulness: Codes are meaningful. Code values should reflect the characteristics of the coded entities, such as mnemonic features unless such a procedures results in inconsistency and inflexibility.

10. Operatability: The code is adequate for present and anticipated data processing both for machine and human use. Care is taken to minimize the clerical effort and computer time required for continuing the operation.

3.4.2. Table Specifications

Table Description / Specification:

Table 1: CompanyInfo_M_T (Company Information Master Table)

Serial No	Attribute	Data Type	Size	Const.	Remarks
1	CompanyName	Varchar	24	PK	Company Name
2	Address1	Varchar	100	Not null	Address
3	City	Varchar	24		City
4	PinCode	Numeric	6		Pin code
5	State	Varchar	24		State
6	Country	Varchar	24		Country
7	Phone	Varchar	24		Phones
8	Fax	Varchar	24		Fax
9	Email	Varchar	48		Email

Table 2: EmpCategory_M_T (Employee Category Master Table)

Serial No	Attribute	Data Type	Size	Const.	Remarks
1	EmpCatCode	Varchar	3	Not null, PK	Employee category code
2	EmpCategory	Varchar	24	Not null, Unique	Employee category

Table 3: Employee_M_T (Employee Master Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	EmpCode	Varchar	6	Not null, PK	Employee Code
2	EmpName	Varchar	24	Not null	Employee Name
3	Address	Varchar	100		Address
4	City	Varchar	24		City
5	PinCode	Numeric	6		Pin code
6	State	Varchar	24		State
7	Country	Varchar	24		Country
8	Phone	Varchar	24		Phones
9	Email	Varchar	48		Email
11	Incharge	Varchar	6	FK Employee_M_T (EmpCode)	Superior Code
12	EmpCatCode	Varchar	3	FK to EmpCategory_M_T	Employee category code

Table 4: User_M_T (User Master Table)

Serial	Attribute	Data Type	Size	Constraint	Remarks

No					
1	UserName	Varchar	24	Not null	User name
2	Password	Varchar	24	Not null	Password
3	EmpCode	Varchar	6	Not null, FK to Employee_M_T	Employee code

Table 5: UserPrivileges_M_T (User Privileges Master Table)

Serial No	**Attribute**	**Data Type**	**Size**	**Const.**	**Remarks**
1	User_Name	Varchar	24	PK	User Name
2	FormID	Varchar	24	PK	Form name
3	View_Privileges	Boolean			Privileges to view the form
4.	Edit_Privileges	Boolean			Privileges to Edit the Record
5	Add_Privileges	Boolean			Privileges to add new record
6	Delete_Privileges	Boolean			Privileges to Delete Record

Table 6: VehicleType_M_T (Vehicle Type Master Table)

Serial No	**Attribute**	**Data Type**	**Size**	**Const.**	**Remarks**
1	VehicleTypeCode	Varchar	3	PK	Vehicle type code
2	VehicleType	Varchar	24	Not Null, unique	Vehicle type(eg. Four Wheeler with A/c / Non A/c / etc)

Table 7: VehicleManufacturer_M_T (Vehicle Manufacturer Master Table)

Serial No	**Attribute**	**Data Type**	**Size**	**Const.**	**Remarks**
1	ManufacturerCode	Varchar	3	PK	Vehicle manufacturer code
2	ManufacturerName	Varchar	24	Not Null, unique	Vehicle manufacturer name

Table 8: Vehicle_M_T (Vehicle Master Table)

Serial No	**Attribute**	**Data Type**	**Size**	**Const.**	**Remarks**
1	VehicleID	Varchar	12	PK	Vehicle Number
2	VehicleTypeCode	Varchar	3	Not null, FK to VehicleType_M_T	Vehicle type code
3	ManufacturerCode	Varchar	3	Not null, FK to VehicleManufacturer_	Manufacturer code

				M_T	
4	ManufacturedIn	Numeric	4		Manufactured Year
5	ChassisNo	Varchar	24		Chassis number

Table 9: Vendor_M_T (Vendor Master Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	VendorId	Varchar	6	PK	Vendor Id
2	Vendor Name	Varchar	24	Not null	Vendor Name
3	Address	Varchar	100	Not null	Address
4	City	Varchar	24		City
5	PinCode	Numeric	6		Pin Code
6	State	Varchar	24		State
7	Country	Varchar	24		Country
8	Phone	Varchar	24		Phone
9	Mobile	Varchar	24		Mobile
10	Fax	Varchar	24		Fax
11	Email	Varchar	48		Email

Table 10: Spares_M_T (Spares Master Table)

Serial No	Attribute	Data Type	Size	Const.	Remarks
1	ItemCode	Varchar	6	PK	Spare part code
2	ItemDesc	Varchar	24	Not null	Spare part description
3	UOM_Code	Varchar	3	Not null	Unit of measure (FK – UOM_M_T(UOM_Code)
4	Rate	Number	10,2	Not null	Rate
5	ROL	Number	10,2	Not null	Reorder level
6	EOQ	Number	10,2		Economic order level
7	MinQty	Number	10,2		Minimum level
8	MaxQty	Number	10,2		Maximum level

Table 11: Service_M_T (Service Master Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	ServiceID	Varchar	6	PK	Service ID
2	ServiceDesc	Varchar	24	Not null	Service Name
3	Rate	Number	10,2		Rate or service

Table 12: SparesInward_M_T (Spares Inward Master Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	InwardNo	Varchar	6	PK	Inward numbser
2	InwardDate	Date		Not null	Inward date
3	VendorId	Varchar	6	FK-Vendor_M_T, Not Null	Vendor Code

4	AuthBy	Varchar	6	FK Employee_M_T (EmpCode)	Authorized Employee
5	RefNo	Varchar	24		Reference number like Invoice No.
6	Remarks	Varchar	100		Remarks

Table 13: SparesInward_T_T (Spares Inward Transaction Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	InwardNo	Varchar	6	FK to SparesInward_M_T	Inward number
2	ItemCode	Varchar	6	PK, FK to Spares_M_T	Spare part code
3	Qty	Number	10,2	Not null	Quantity

Table 14: SparesOutward_M_T (Spares Outward Master Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	OutwardNo	Varchar	6	PK	Outward number
2	OutwardDate	Date		Not null	Outward date
3	AuthBy	Varchar	6	FK Ref Employee_M_T (EmpCode)	Authorized Employee
4	JobCardNo	Varchar	6	FK JobCard_M_T	Like JobCardNo(s)
5	Remarks	Varchar	100		Remarks

Table 15: SparesOutward_T_T (Spares Outward Transaction Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	OutwardNo	Varchar	6	PK, FK to SparesOutward_M_T	Outward number
2	ItemCode	Varchar	6	PK, FK to Spares_M_T	Spare part code
3	Qty	Number	10,2	Not null	Quantity

Table 16: SparesInv_T_T (Spares Inventory Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	ItemCode	Varchar	10	PK, FK to Spares_M_T	Spare part code
2	Qty	Number	10,2	Not null	Quantity(Current Stock)

Table 17: Customer_M_T (Customer Master Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	CustCode	Varchar	6	PK	Customer code
2	CustName	Varchar	24	Not null	Customer name

3	Address	Varchar	100	Not null	Address
4	City	Varchar	24		City
5	PinCode	Varchar	6		Pin Code
6	State	Varchar	24		State
7	Country	Varchar	24		Country
8	Phone	Varchar	24		Phone
9	Mobile	Varchar	24		Mobile
10	Fax	Varchar	24		Fax
11	Email	Varchar	48		Email
12	VehicleId	Varchar	12	FK(Vehicle_M_T – VehicleId)	Vehicle Number

Table 18: Enquiry_M_T (Enquiry Master Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	SerialNo	Varchar	6	PK	Enquiry serial number
2	CurrDate	Date	24	Not null	Current date
4	CustName	Varchar	24		Customer Name
5	CustAddress	Varchar	100		Address of Customer
6	ContactNo	Varchar	24		Contact number
7	EnquiryStatus	Char	1		Enquiry status

Table 19: Appointments_M_T (Appointments Master Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	SerialNo	Varchar	6	PK	Appointment serial number
2	CurrDate	Date		Not null	Current date
3	CustName	Varchar	24		Customer Name
4	Enq_SlNo	Varchar	10	Nullable FK Enquiry_M_T	Enquiry Sl. No.
4	ContactNo	Varchar	100		Contact number
5	AppointmentDate	Date		Not Null	
6	AppointStatus	Char	1	Not Null	Appointment status

Table 20: Quotation_M_T (Quotation Master Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	QuotNo	Varchar	6	Primary Key	Quotation number
2	QuotDate	Date		Not Null	Quotation date
3	CustName	Varchar	24	Not Null	Customer code
4	CustAddress	Varchar	100		Customer Address
5	PreparedBy	Varchar	6	FK Employee_M_T EmpCode	Employee who prepared quote
6	QuotStatus	Char	1	Not Null	Status of the Quot. whether approved or cancelled

Table 21: QuotationService_T_T (Quotation Service Transaction Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	QuotSNo	Number	6	PK, FK to Quotation_T_T	Quotation serial number
2	ServiceID	Varchar	6	PK, FK to Service_M_T	Service ID
3	Amount	Number	10,2		Service amount

Table 22: QuotationSpares_T_T (Quotation Spares Transaction Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	QuotSNo	Number	6	PK, FK to Quotation_T_T	Quotation serial number
2	ItemCode	Varchar	6	PK, FK to Spares_M_T	Spares code
3	Quantity	Number	5,3	Not null	Quantity
4	Amount	Number	10,2	Not null	Amount

Table 23: JobCard_M_T (Job Card Master Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	JobCardNo	Number	6	PK	Jobcard number
2	CurrDate	Date		Not null	Current date
3	QuotNo	Varchar	24	FK to Quotation_M_T	Quotation number
4	CustCode	Varchar	10	FK to customer_M_T	Customer code
5	VehicleID	Varchar	10	FK to vehicle_M_T	Vehicle code
6	ExecutionDate	Date		Not Null	Date of Work
7	Status	Char	1	Initialized, Process, Complete, Cancelled	Job status
8	PreparedBy	Varchar	6	FK Employee_M_T (EmpCode)	Employee who prepared quote

Table 24: JobCard_Emp_T_T (Job Card Employee Transaction Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	JobCardNo	Number	6	PK, FK to JobCard_M_T	Jobcard number
2	EmpCode	Varchar	5	PK,FK to Employee_M_T	Employee code (technician)

Table 25: Invoice_M_T (Invoice Master Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks

1	InvoiceNo	Number	6	PK	Invoice number
2	CurrDate	Date	24	Not null	Current date
3	JobCardNo	Varchar	6	FK to JobCard_M_T	Job Card No
4	CustCode	Varchar	10	FK to customer_M_T	Customer code
5	VehicleID	Varchar	10	FK to vehicle_M_T	Vehicle code
6	PreparedBy	Varchar	24		Employee who prepared quote
7	Tax	Varchar	24		Tax name if applicable
8	Tax_Amt	Numeric	10,2		Amount of tax
9	Remarks	Varchar	100		Remarks
10	Revisit_Dt	Date		Nullable	
11	RevisitPurpose	Varchar	100	Nullable	

Table 26: InvoiceService_T_T (Invoice Service Transaction Table)

Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	InvoiceNo	Number	6	PK, FK to Invoice_M_T	Invoice number
2	ServiceID	Varchar	6	PK, FK to Service_M_T	Service ID
3	Amount	Number	10,2	Not Null	Service amount

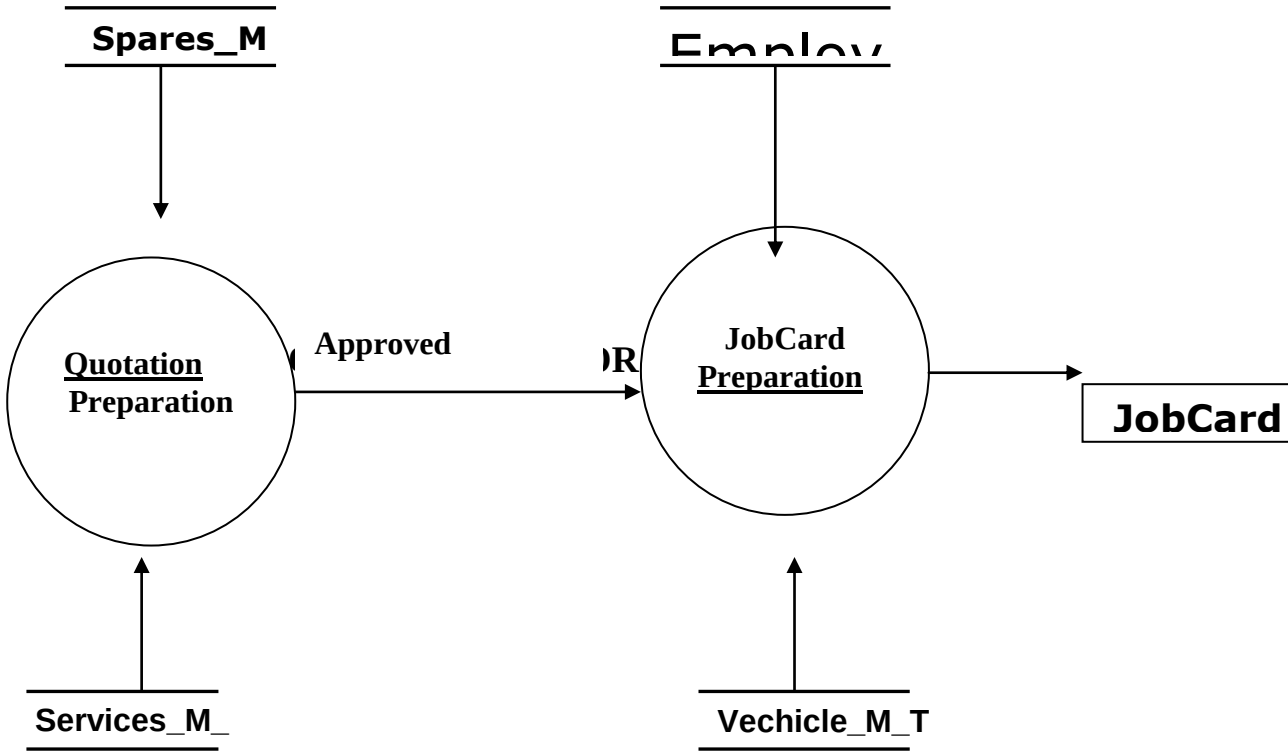
Table 27: InvoiceSpares_T_T (Invoice Spares Transaction Table)

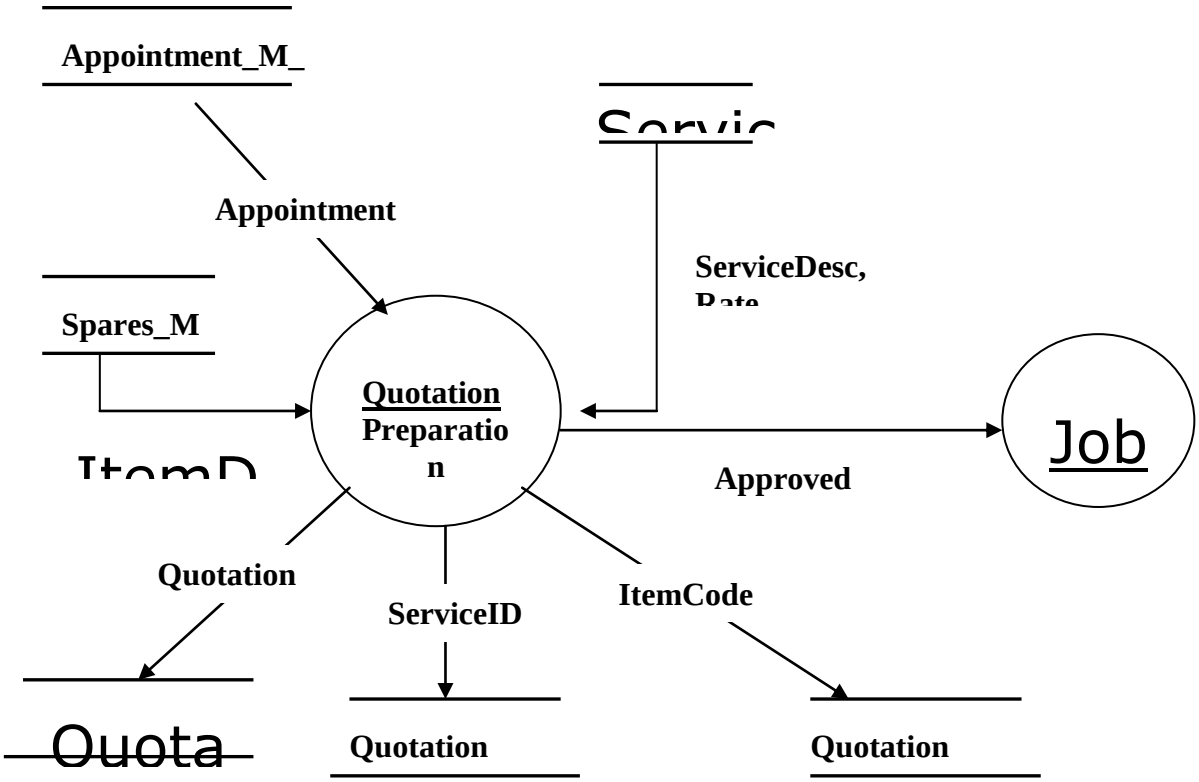
Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	InvoiceNo	Number	6	PK, FK to Invoice_M_T	Invoice number
2	ItemCode	Varchar	6	PK, FK to Spares_M_T	Spares code
3	Quantity	Number	5,3	Not null	Quantity
4	Amount	Number	10,2	Not null	Amount

Table 28: UOM_M_T (Unit of Measurement Master Table)

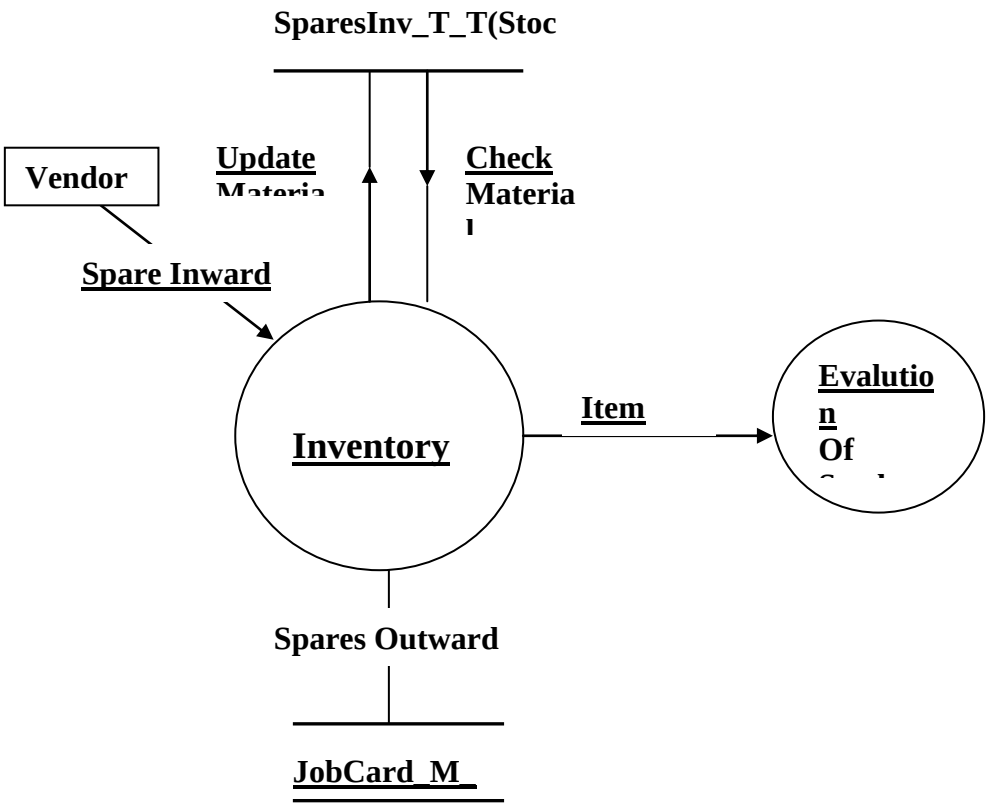
Serial No	Attribute	Data Type	Size	Constraint	Remarks
1	UOM_Code	Number	3	PK	UOM_Code
2	Description	Varchar	24	Not Null	Unit Description

DATAFLOW DIAGRAM FOR JOBCARD

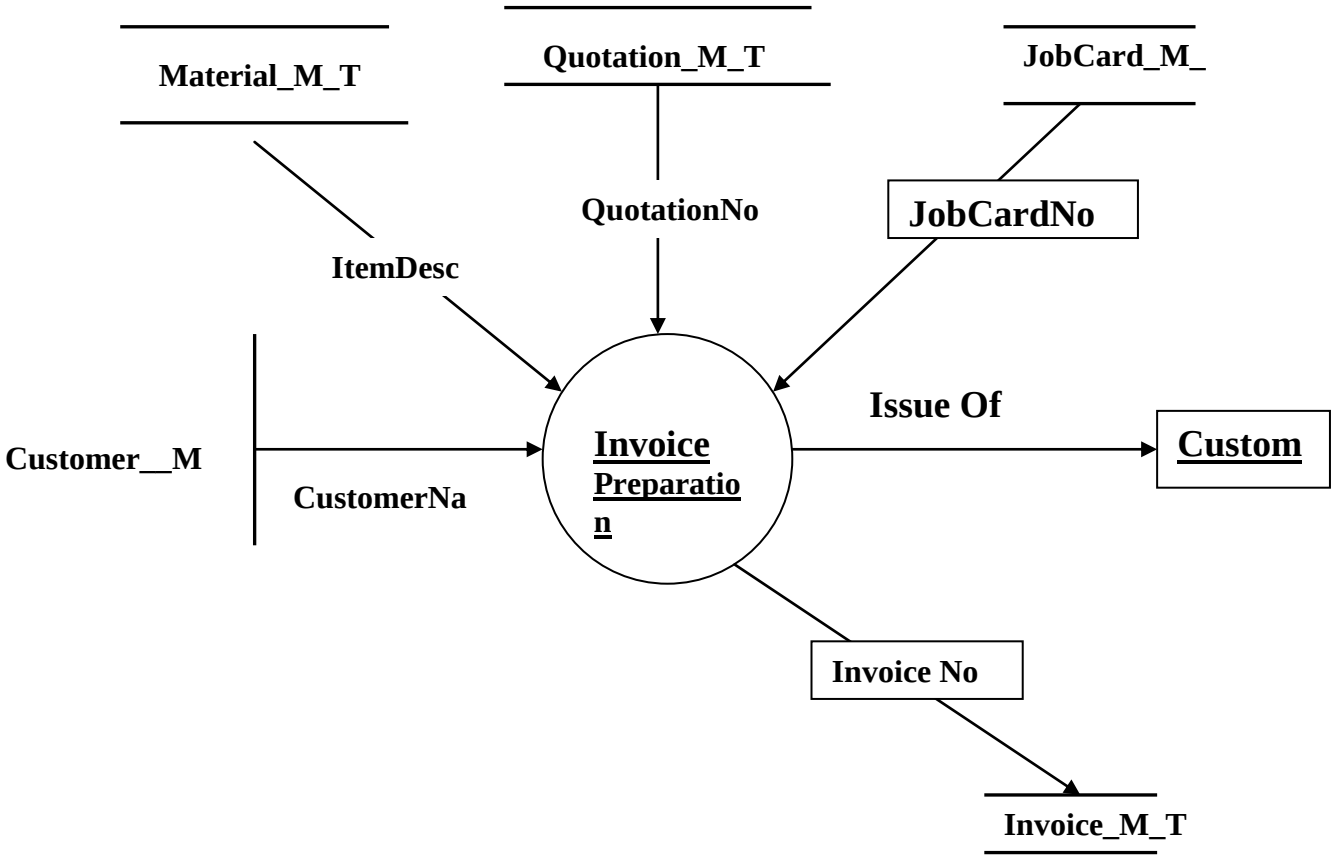



Appointment_M_
Appointment
Spares_M
Quotation Preparation
ServiceDesc,
Job
Approved
Quotation
ServiceID
ItemCode
Quota
Quotation
Quotation

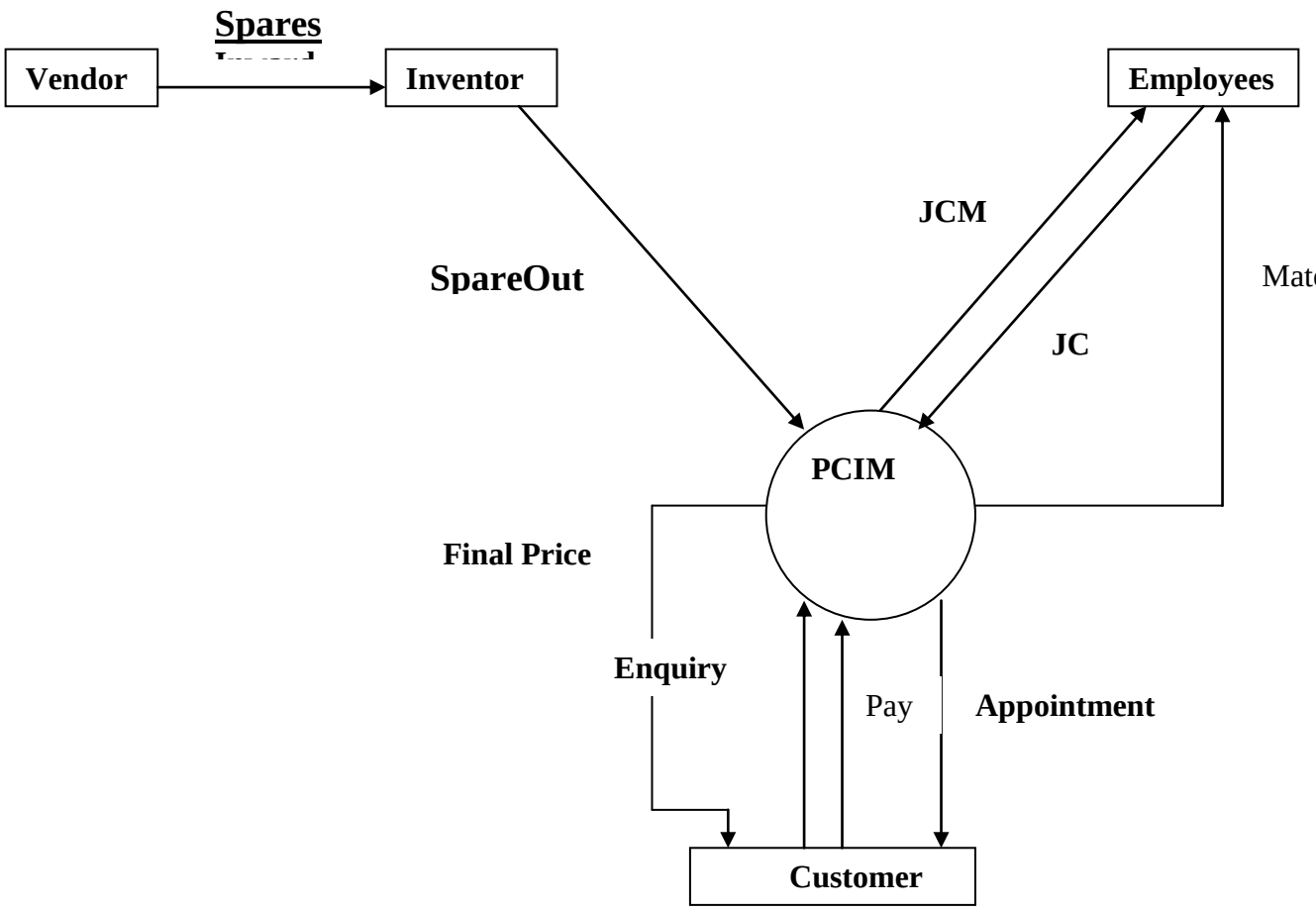
3 DATAFLOW DIAGRAM FOR INVENTORY

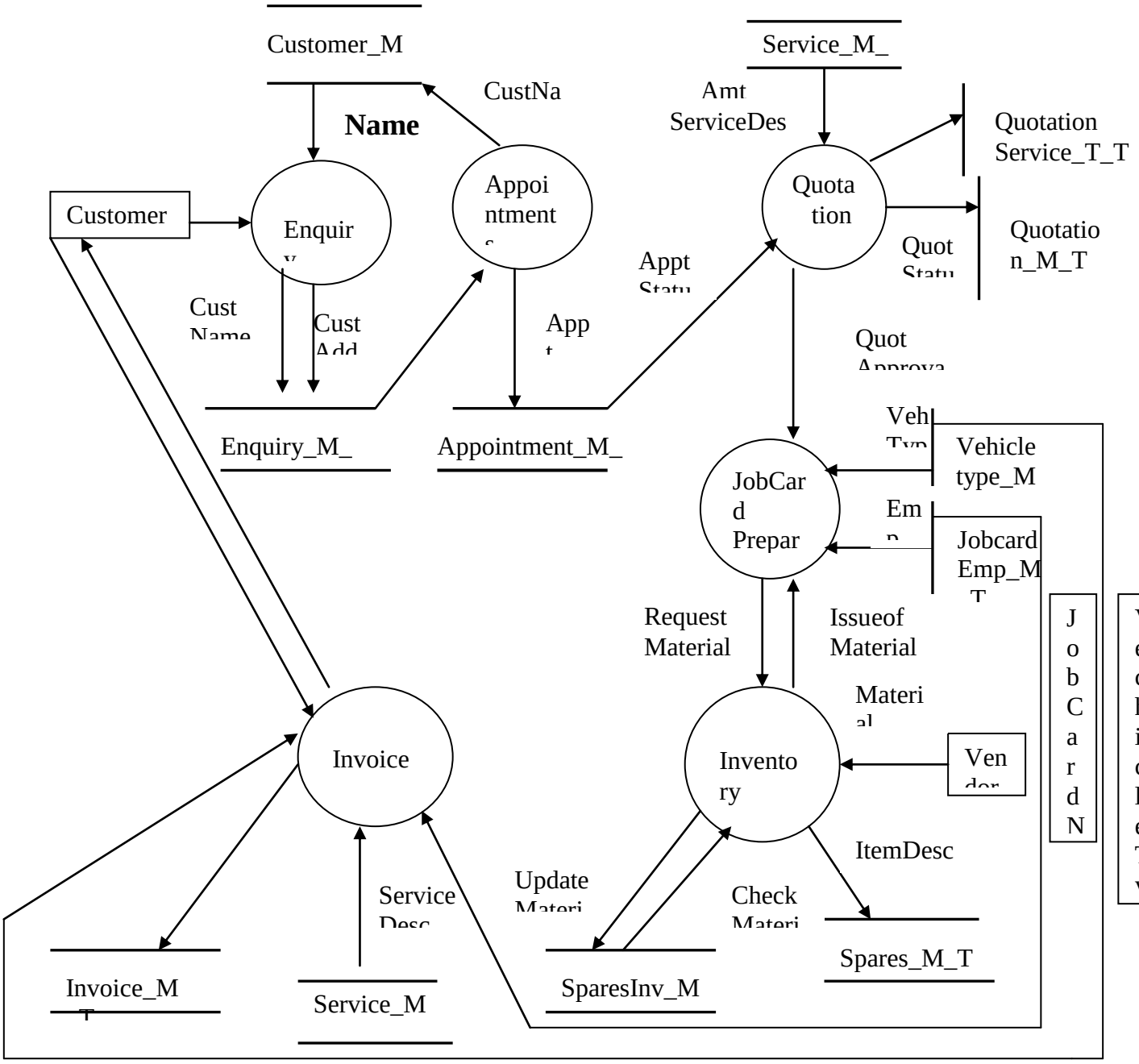


4 DATAFLOW DIAGRAM FOR INVOICE

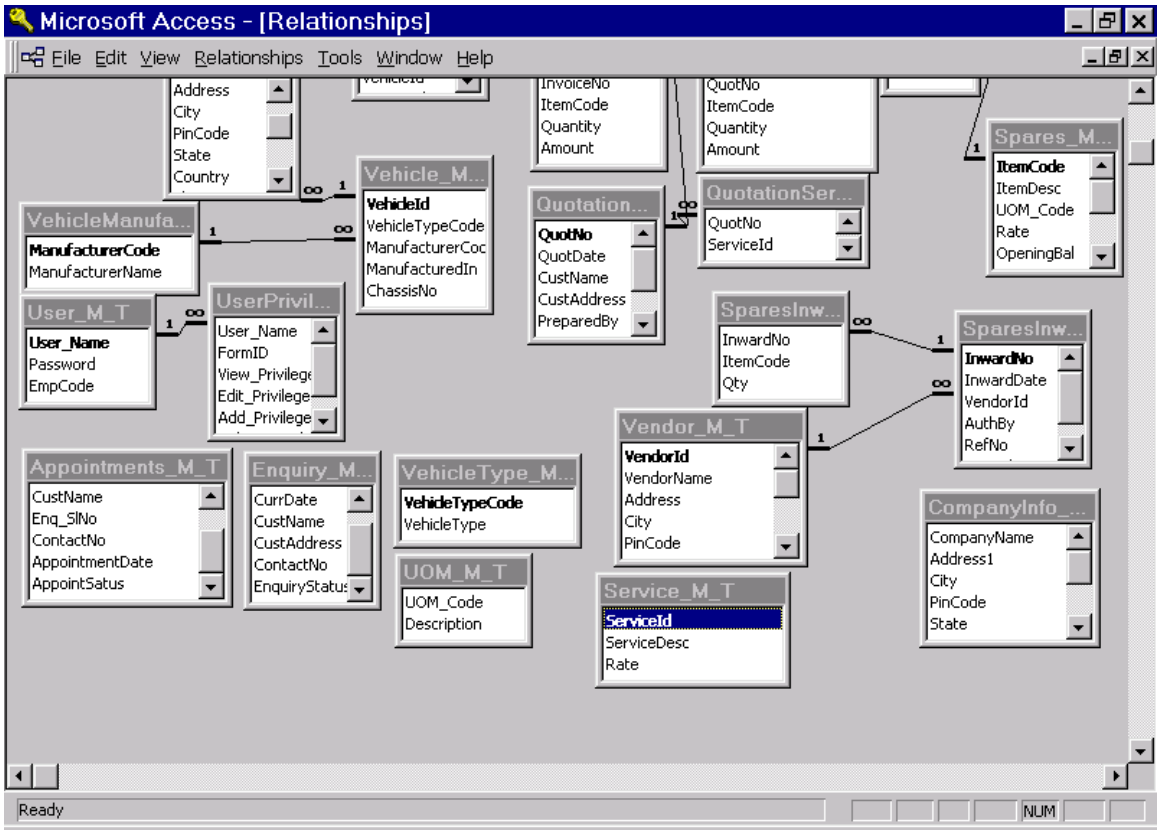


CONTEXT LEVEL DIAGRAM

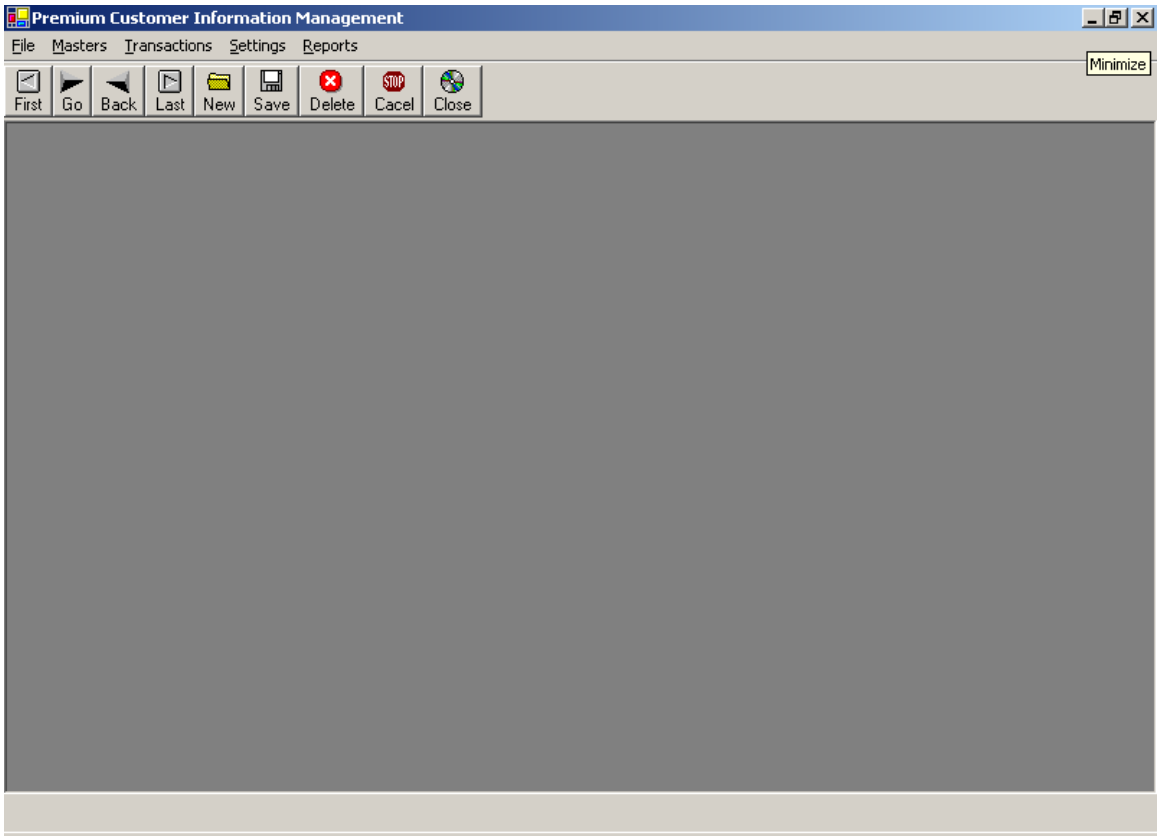



Customer_M
Service_M_
Name
CustNa
Amt
ServiceDes
Quotation
Service_T_T
Customer
Enquir
Appoi
ntment
Quota
tion
Quotatio
n_M_T
Quot
Cust
Name
Cust
Add
App
t
Appt
Quot
Enquiry_M_
Appointment_M_
Veh
Vehicle
type_M
JobCar
d
Prepar
Em
Jobcard
Emp_M
T
Request
Material
Issueof
Material
Materi
al
J
o
b
C
a
r
d
N
Invoice
Inventor
y
Ven
dor
ItemDesc
Update
Materi
Service
Desc
Check
Materi
Spares_M_T
Invoice_M
T
Service_M
SparesInv_M

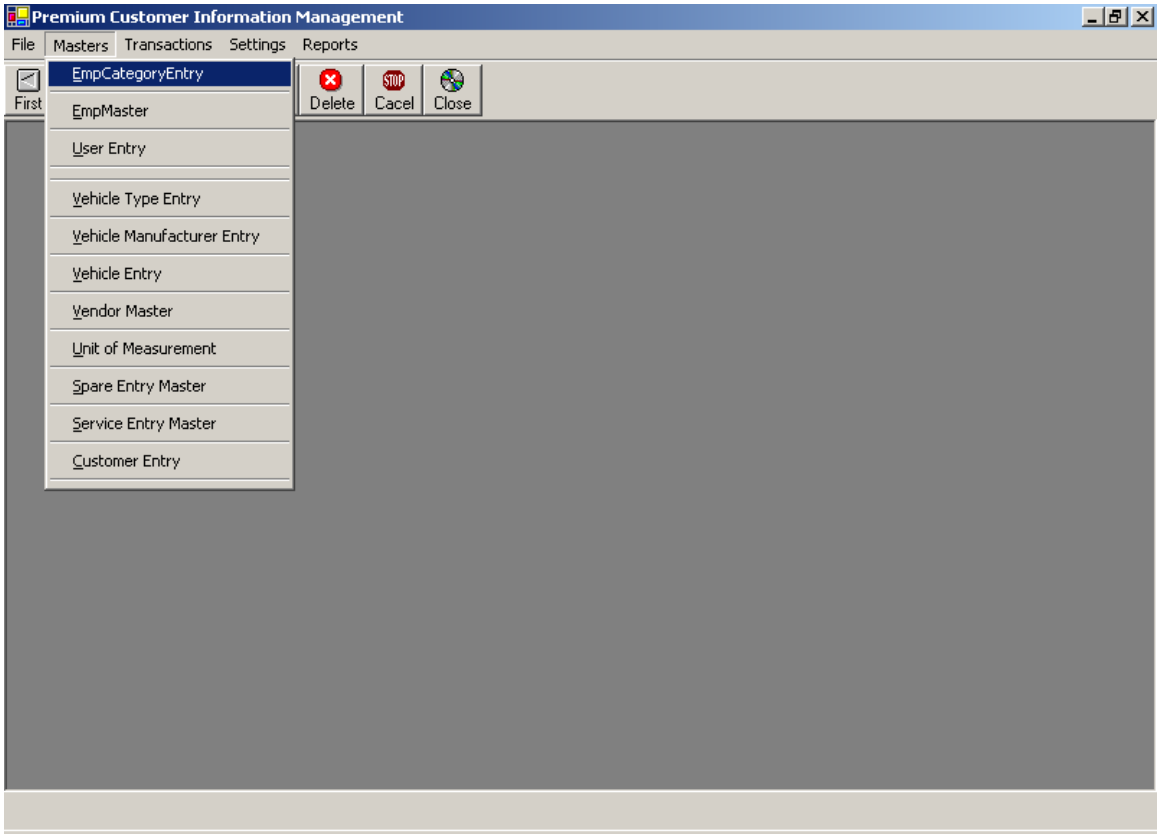
Data Flow Diagrams

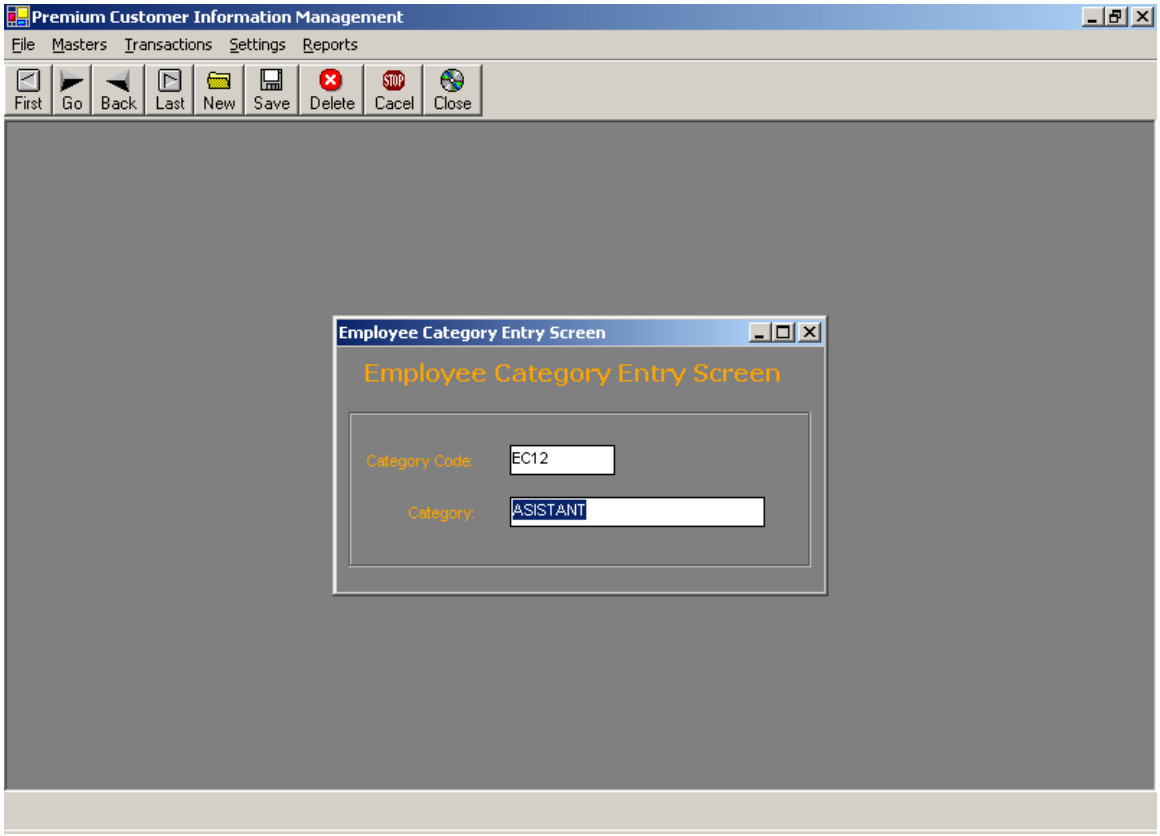


3.8 Input /Output Screens Reports

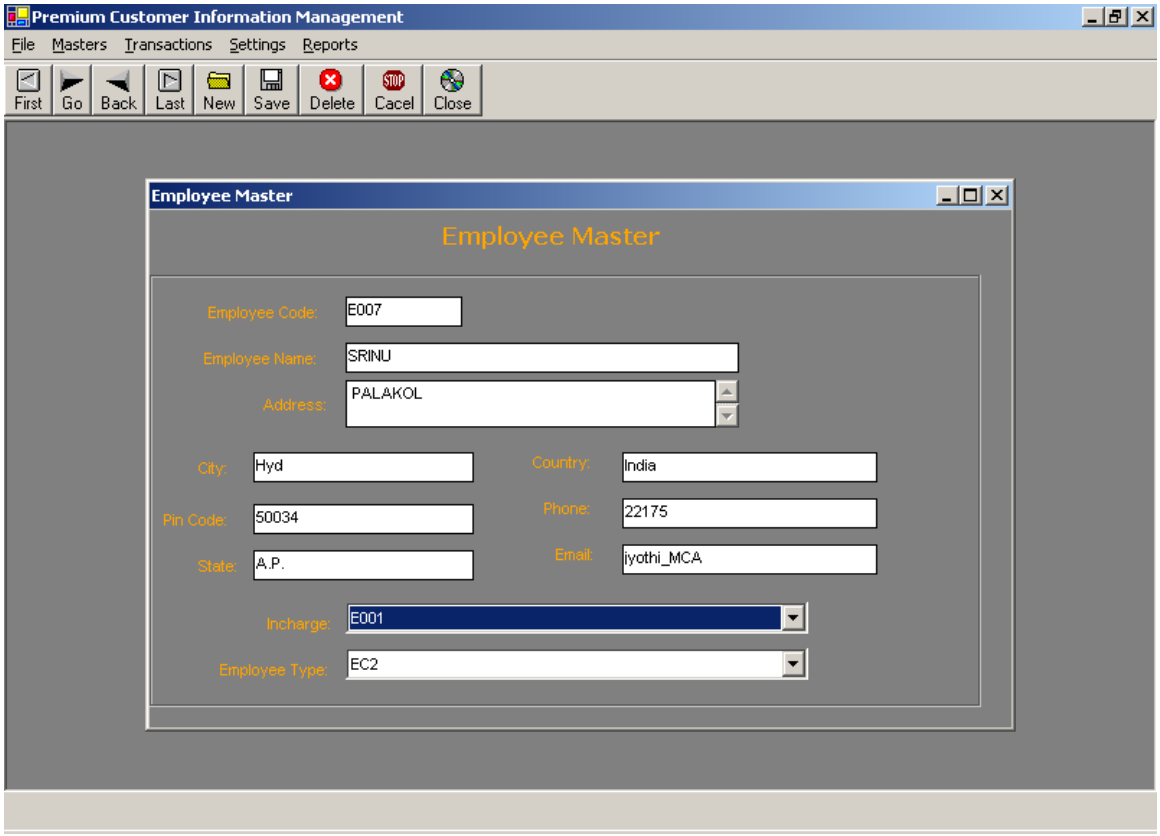


Premium Customer Information Management

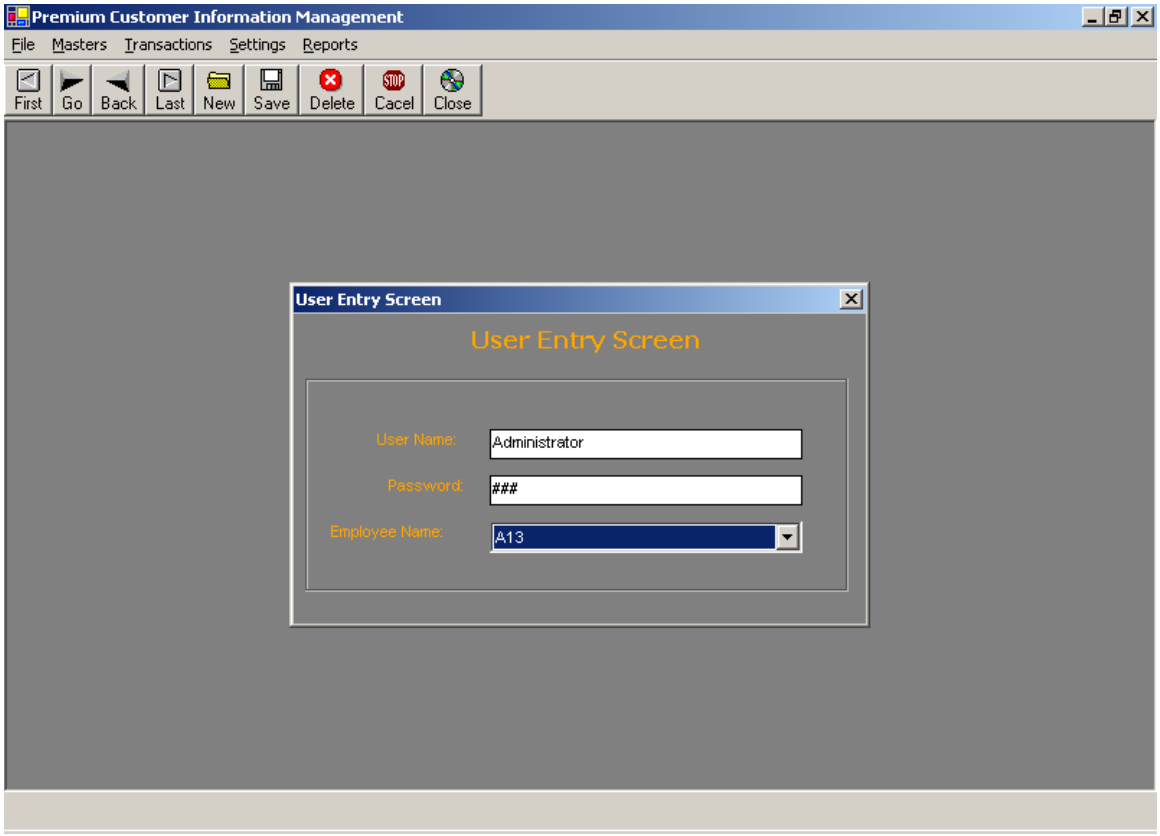


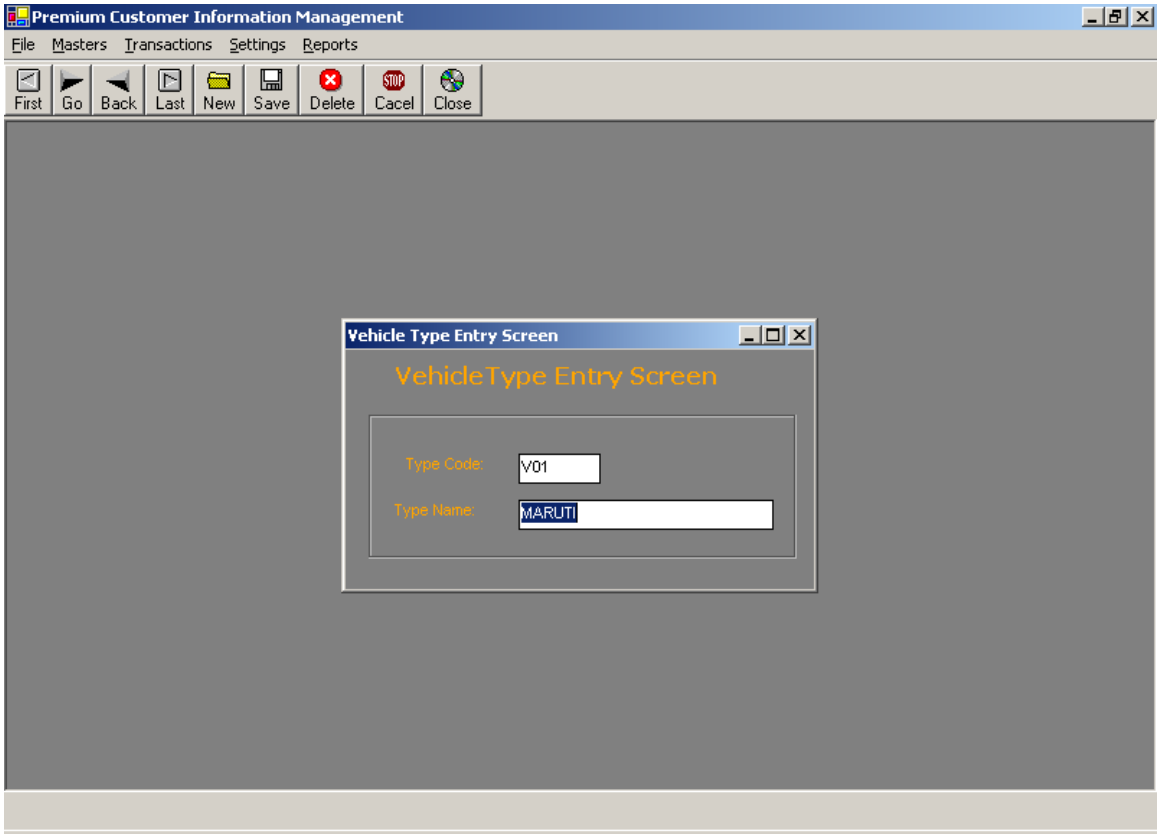

Premium Customer Information Management
File
Masters
Transactions
Settings
Reports
First
Go
Back
Last
New
Save
Delete
Cacel
Close
Employee Category Entry Screen
Employee Category Entry Screen
Category Code:
EC12
Category:
ASISTANT

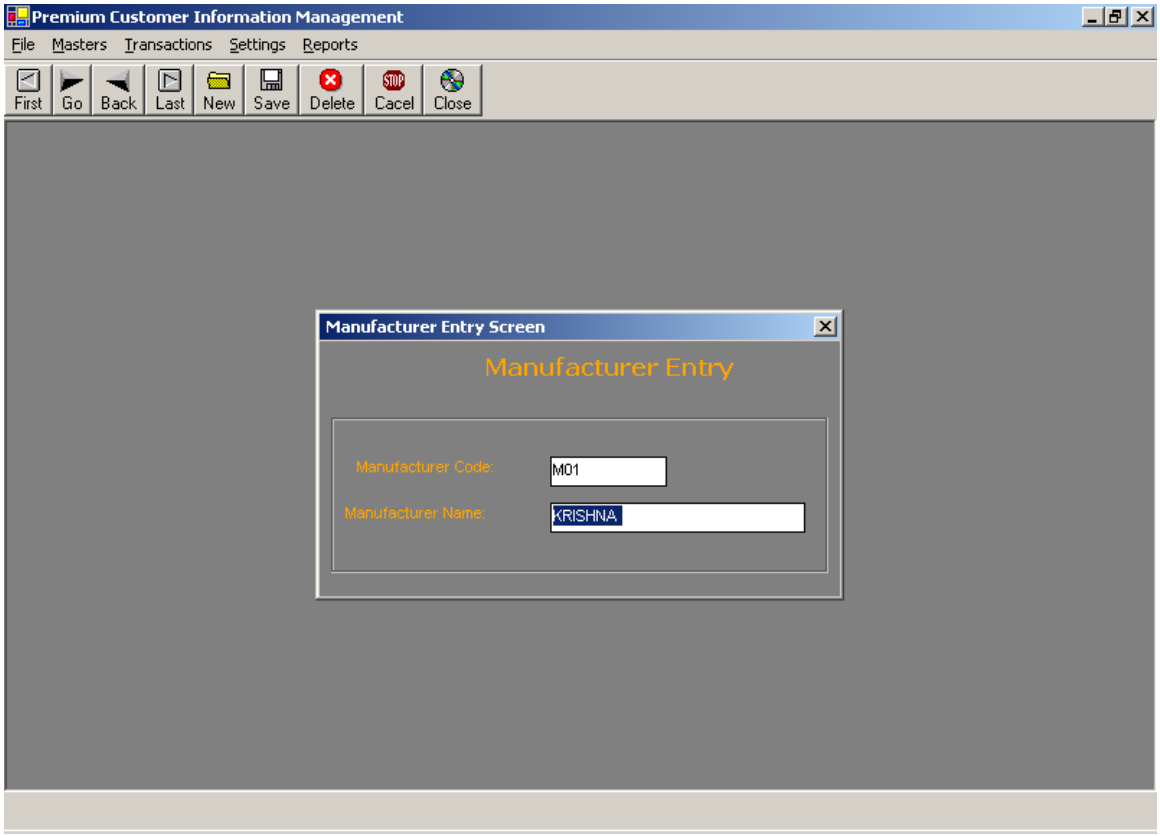
Premium Customer Information Management

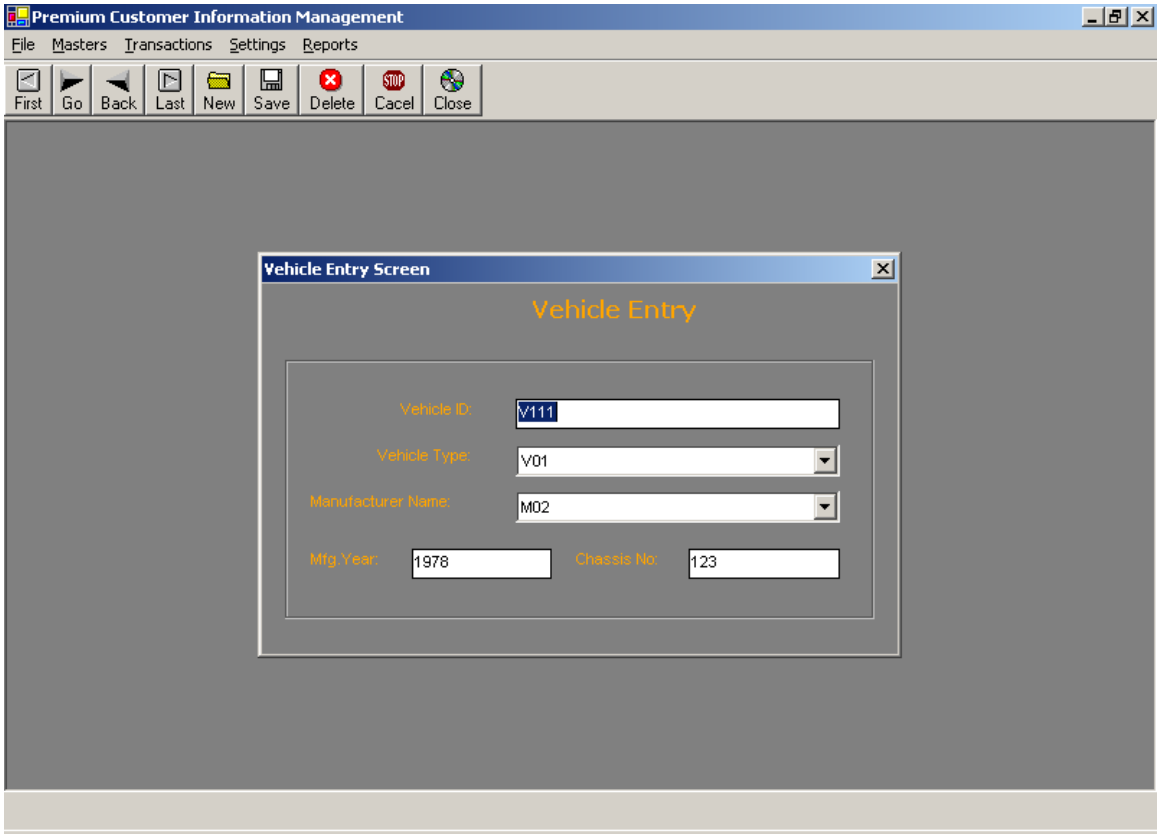


Premium Customer Information Management

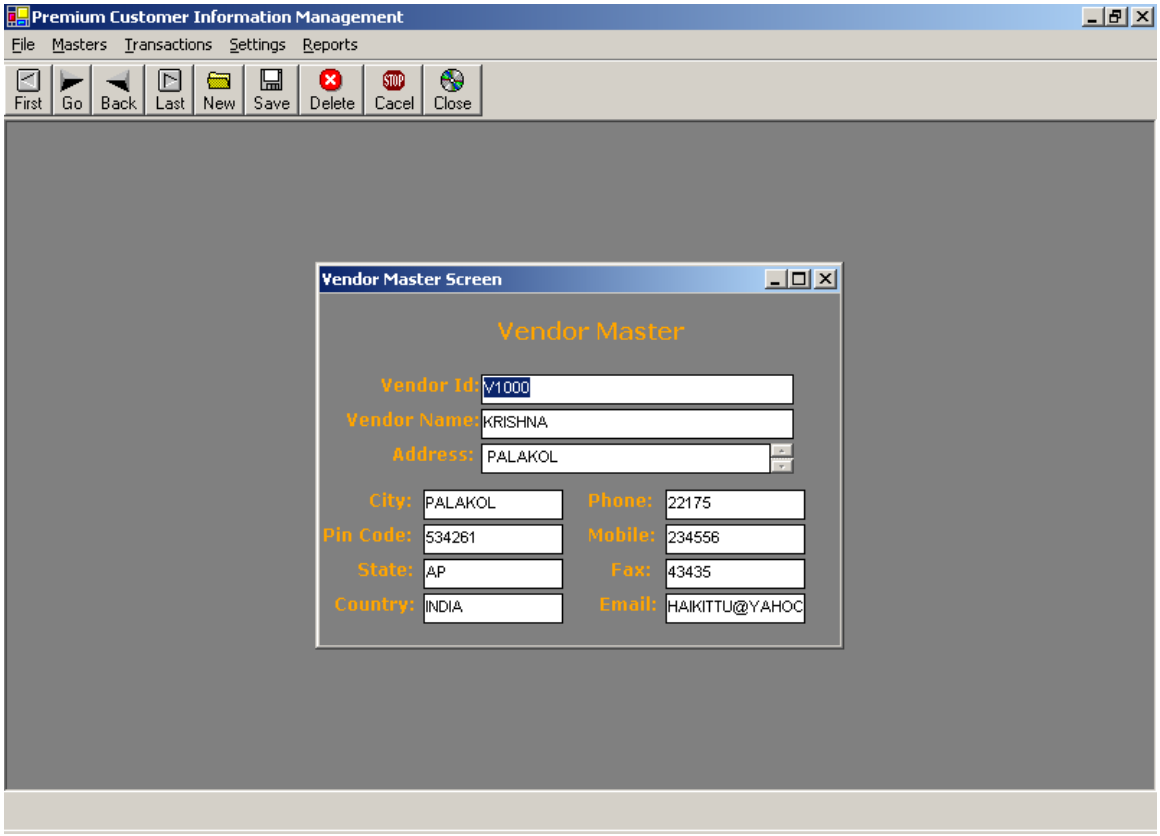


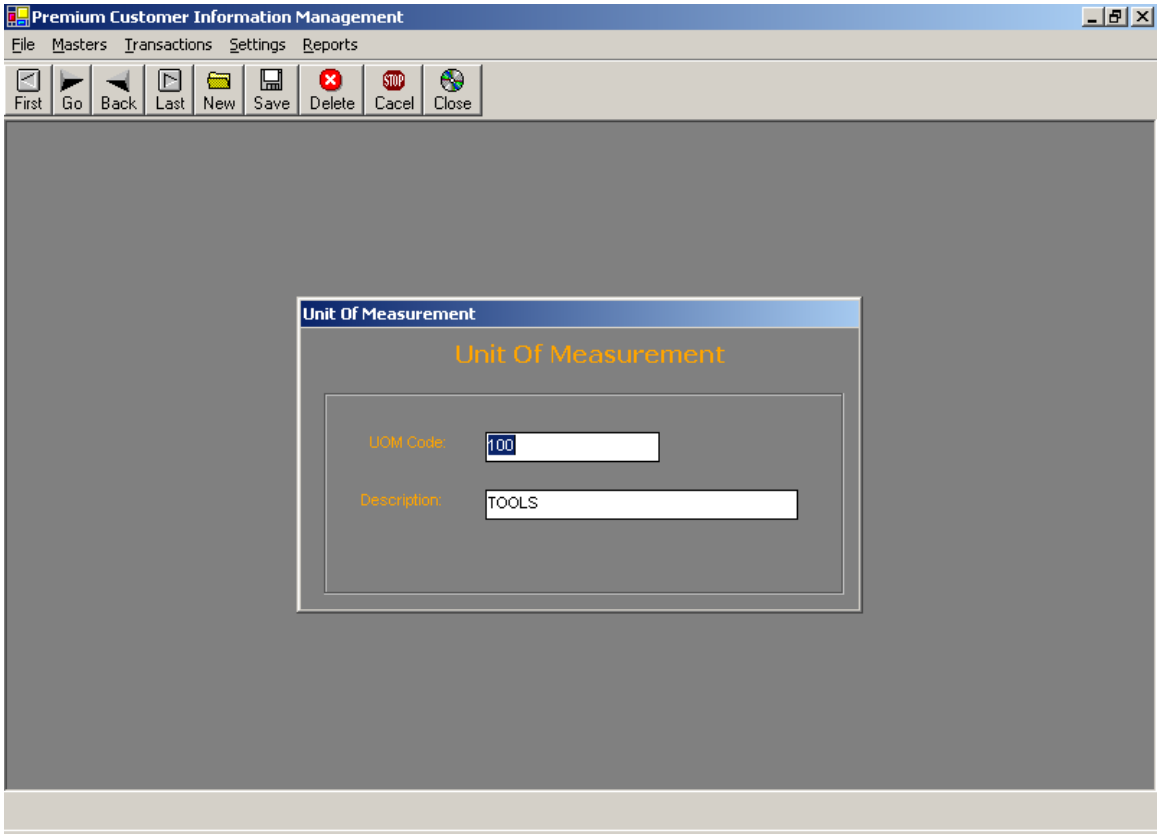

Premium Customer Information Management
File
Masters
Transactions
Settings
Reports
First
Go
Back
Last
New
Save
Delete
Cacel
Close
Vehicle Type Entry Screen
VehicleType Entry Screen
Type Code:
V01
Type Name:
MARUTI


Premium Customer Information Management
File
Masters
Transactions
Settings
Reports
First
Go
Back
Last
New
Save
Delete
Cacel
Close
Manufacturer Entry Screen
Manufacturer Entry
Manufacturer Code:
M01
Manufacturer Name:
KRISHNA

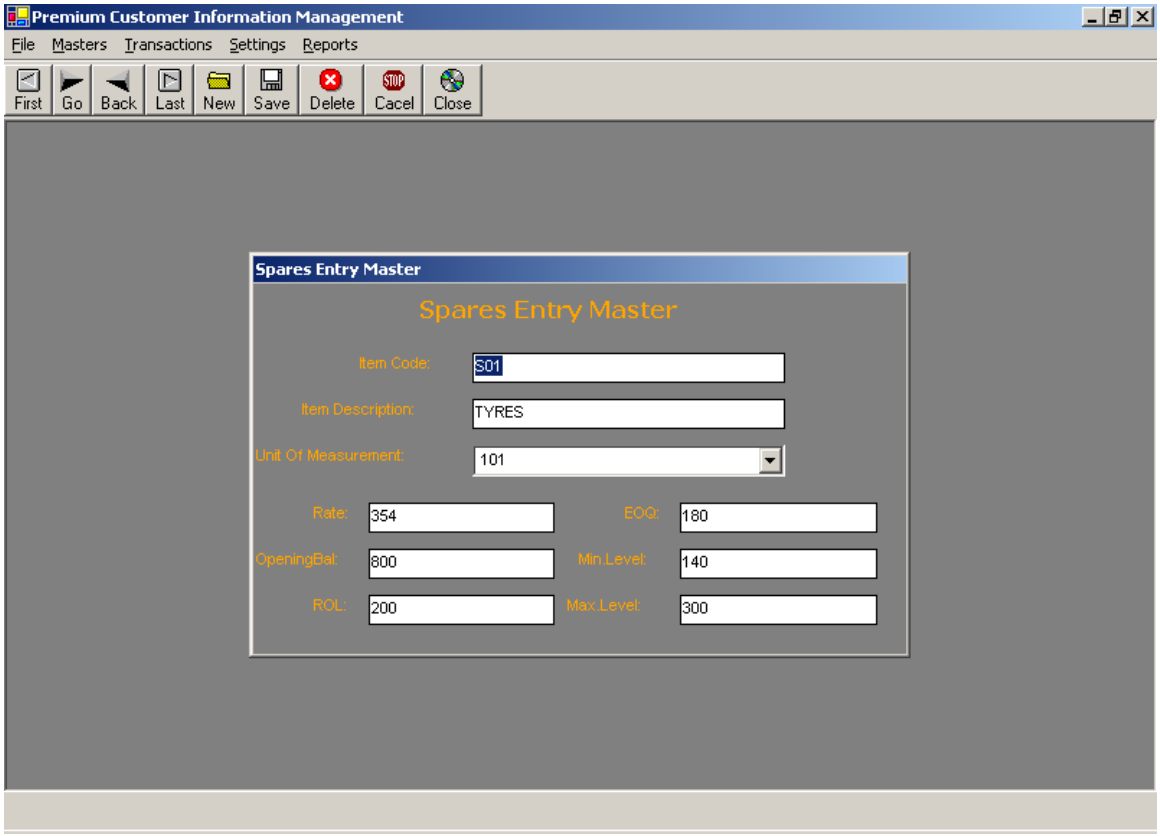

Premium Customer Information Management
File
Masters
Transactions
Settings
Reports
First
Go
Back
Last
New
Save
Delete
Cacel
Close
Vehicle Entry Screen
Vehicle Entry
Vehicle ID:
V111
Vehicle Type:
V01
Manufacturer Name:
M02
Mfg.Year:
1978
Chassis No:
123

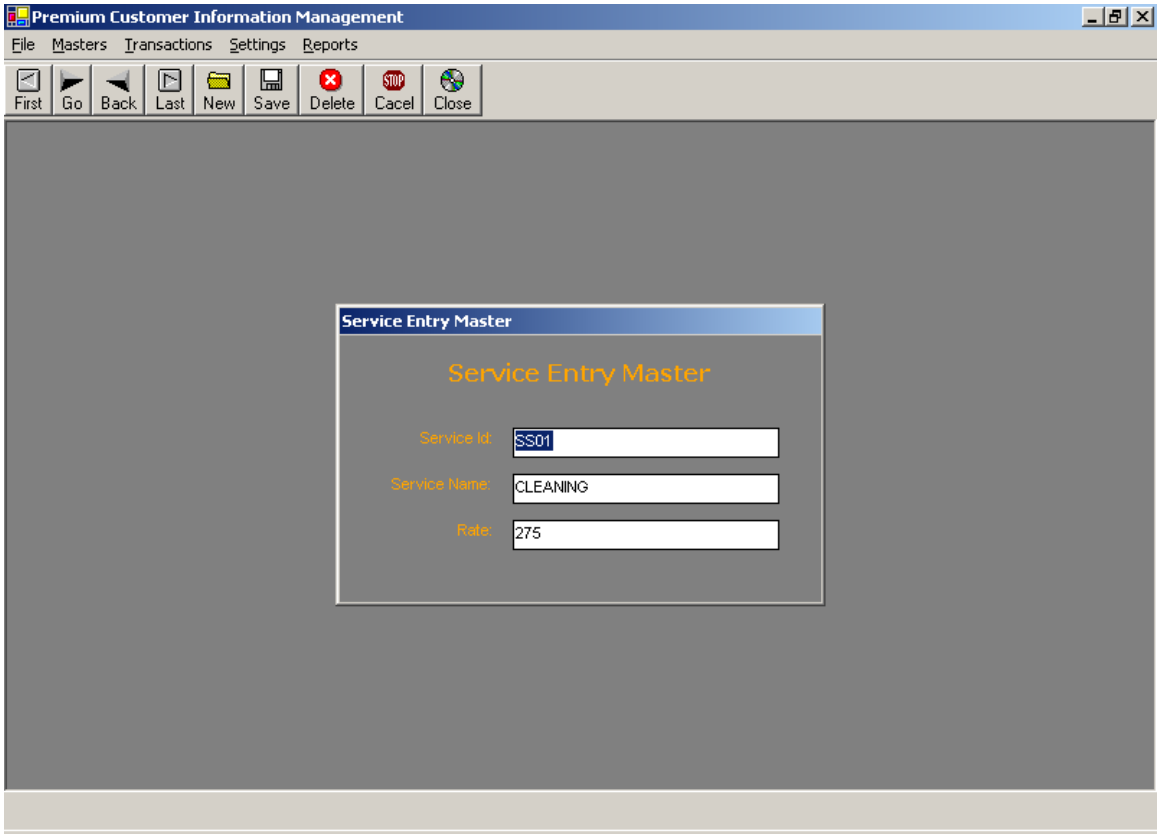
Premium Customer Information Management



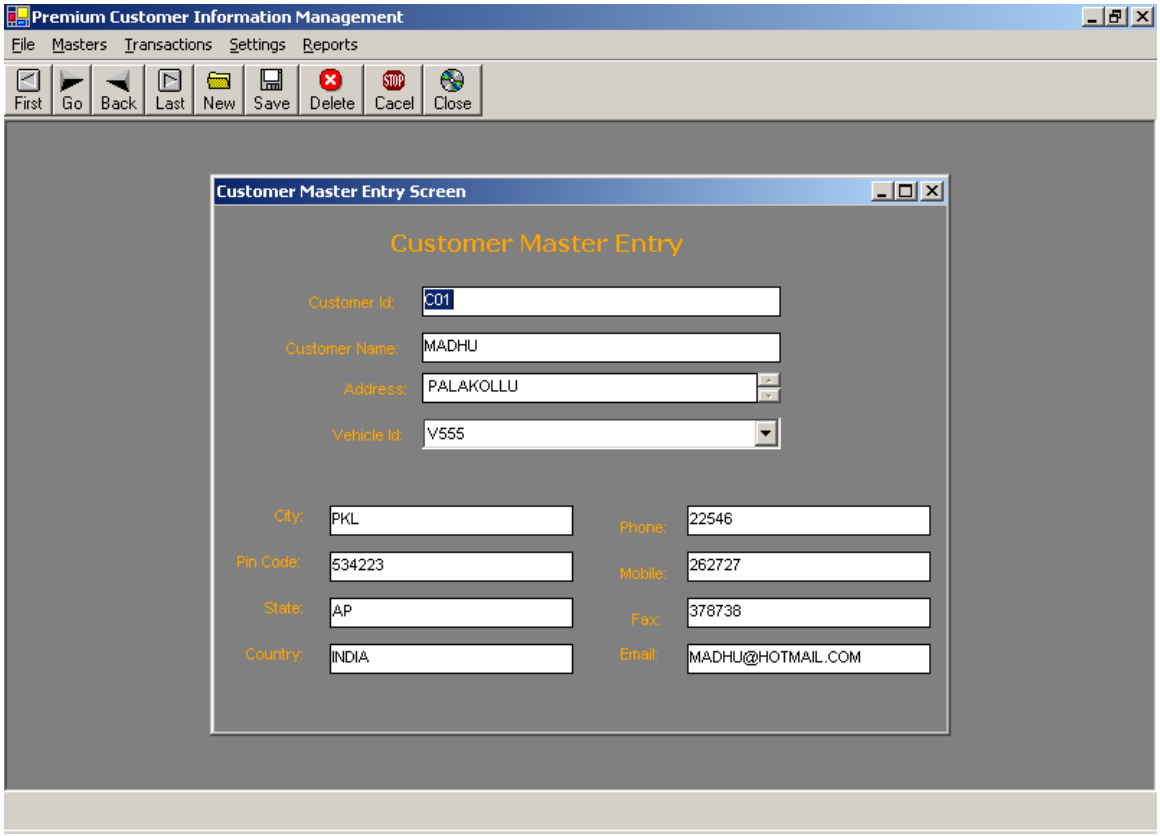

Premium Customer Information Management
File
Masters
Transactions
Settings
Reports
First
Go
Back
Last
New
Save
Delete
Cacel
Close
Unit Of Measurement
Unit Of Measurement
UOM Code:
100
Description:
TOOLS

Premium Customer Information Management

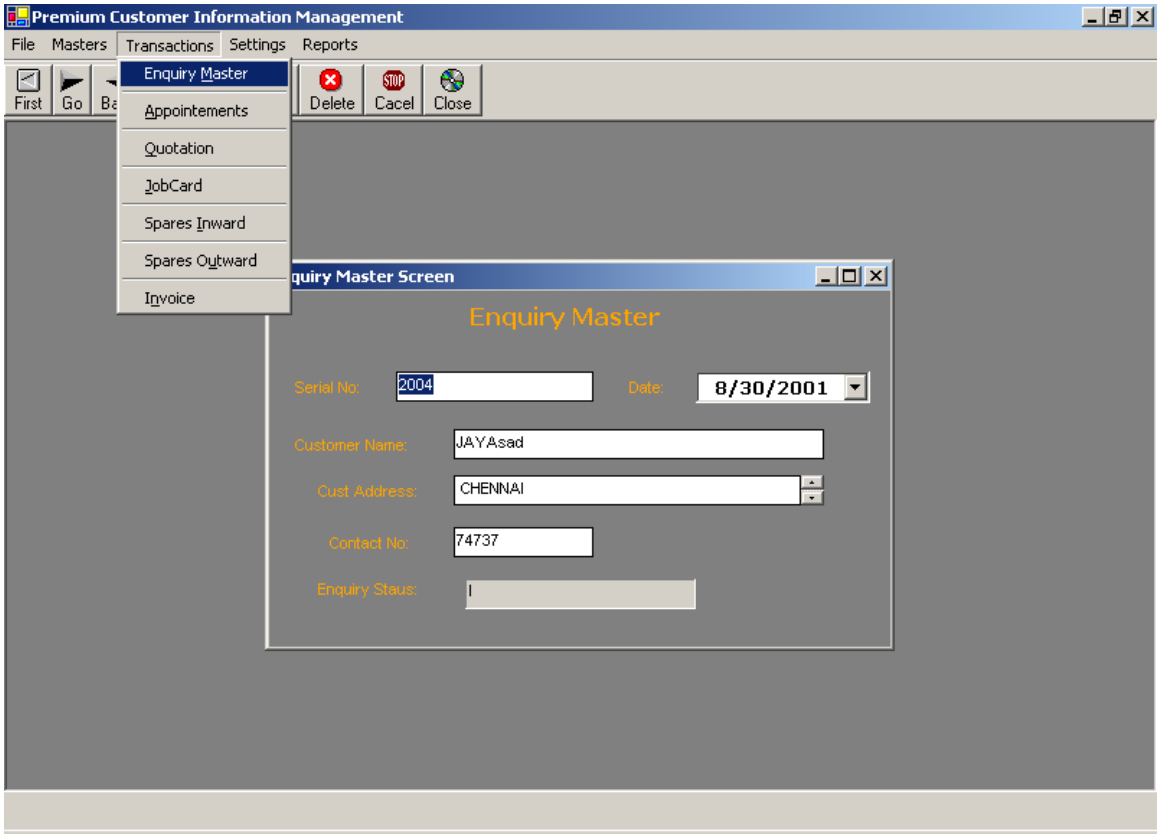



Premium Customer Information Management
File
Masters
Transactions
Settings
Reports
First
Go
Back
Last
New
Save
Delete
Cacel
Close
Service Entry Master
Service Entry Master
Service Id:
SS01
Service Name:
CLEANING
Rate:
275

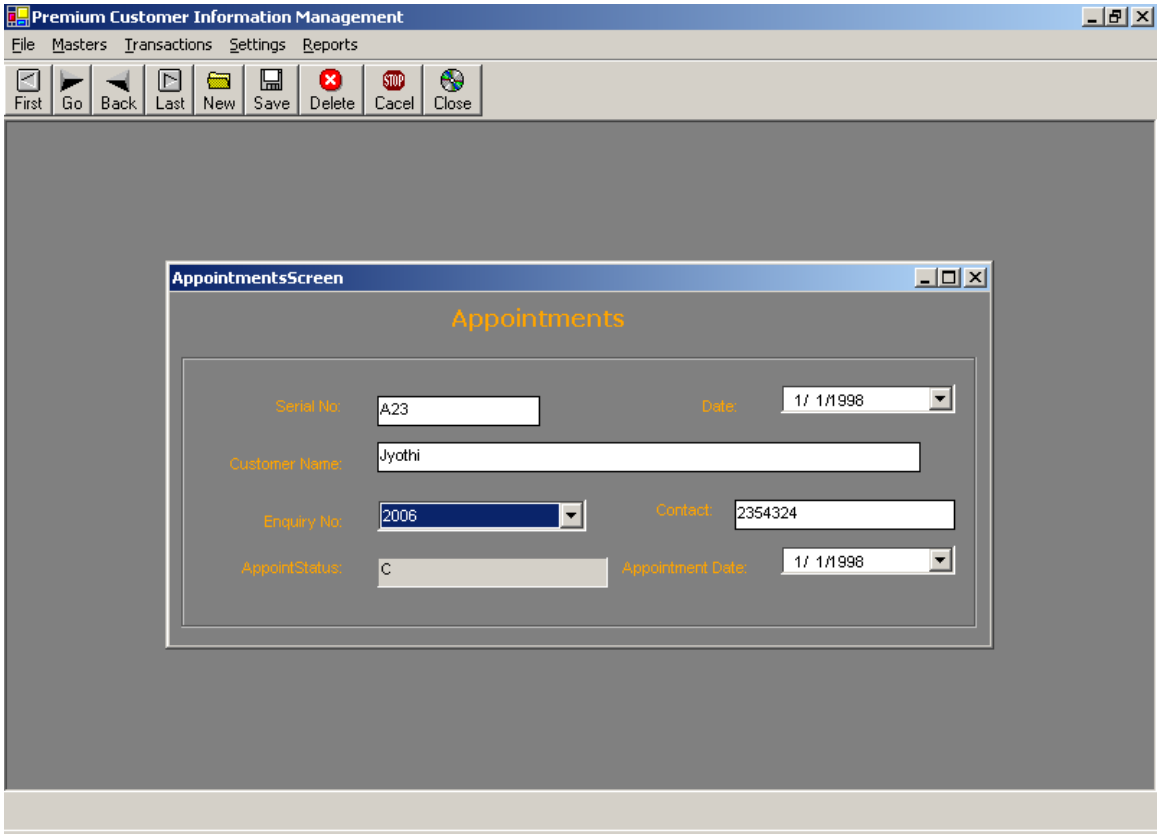
Premium Customer Information Management



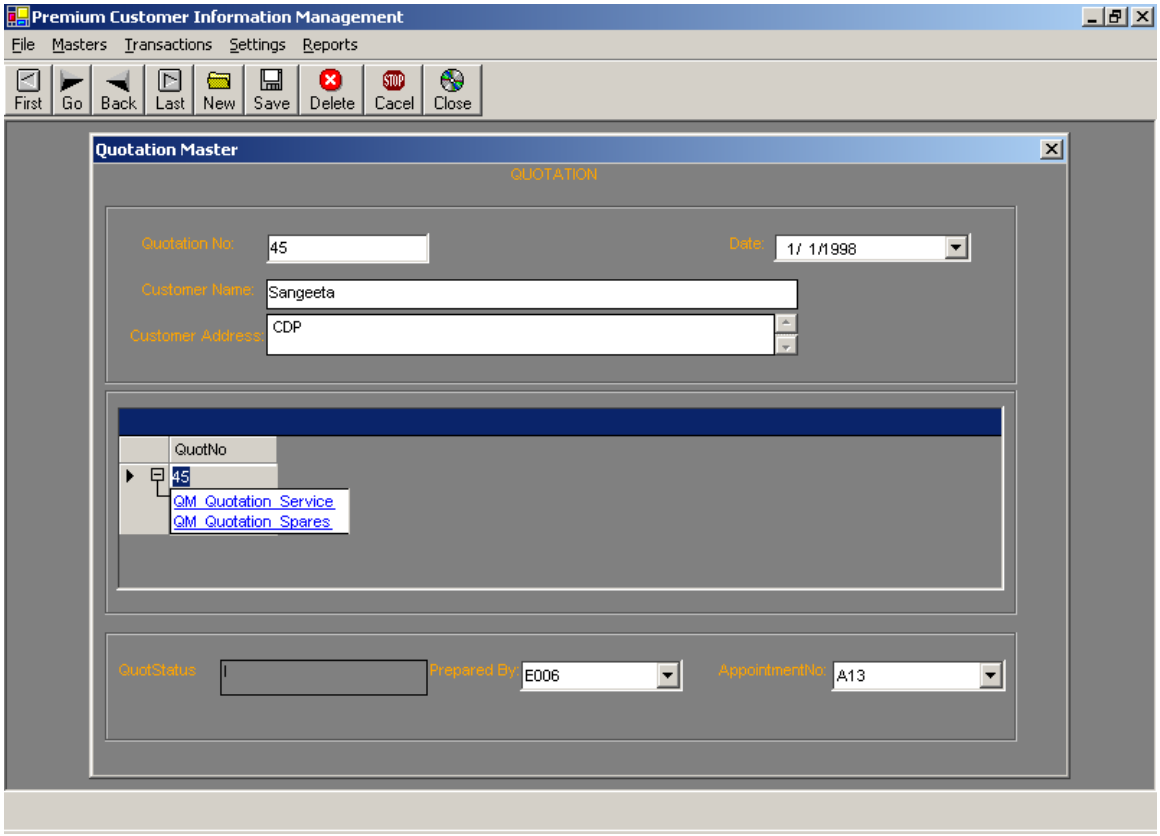
Premium Customer Information Management



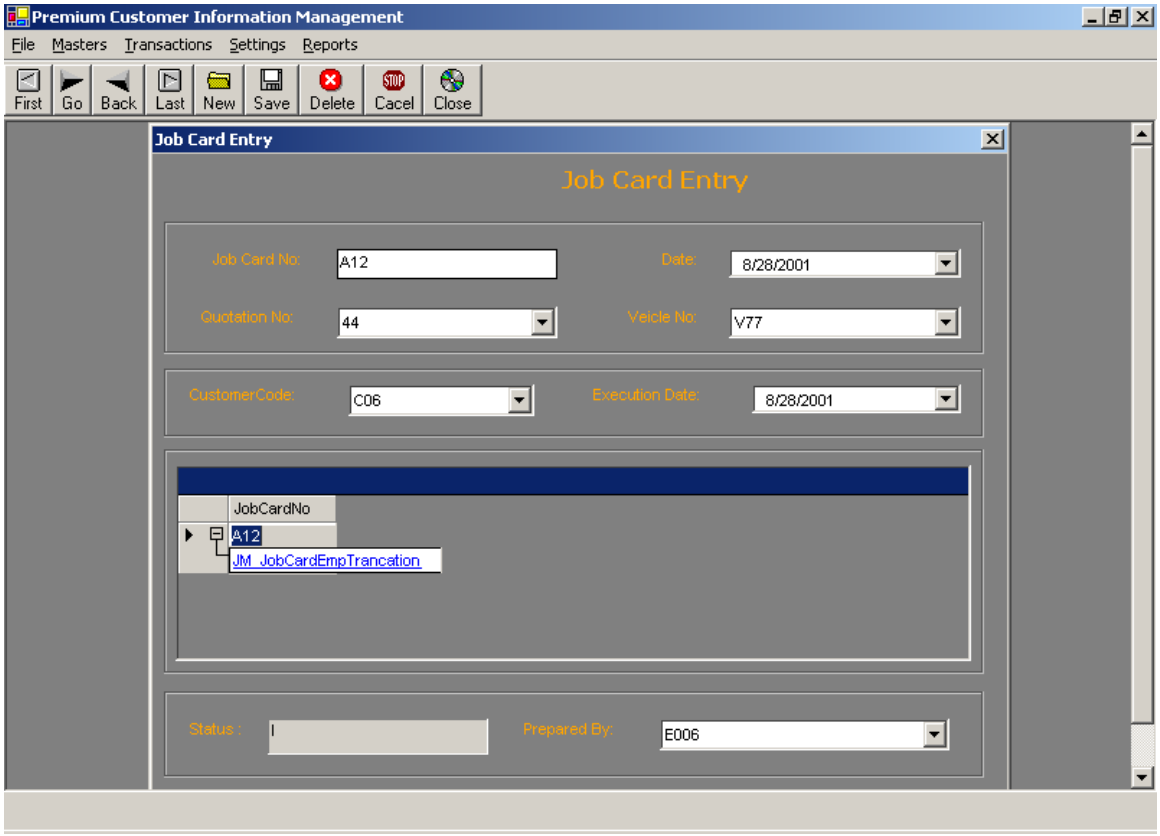
Premium Customer Information Management

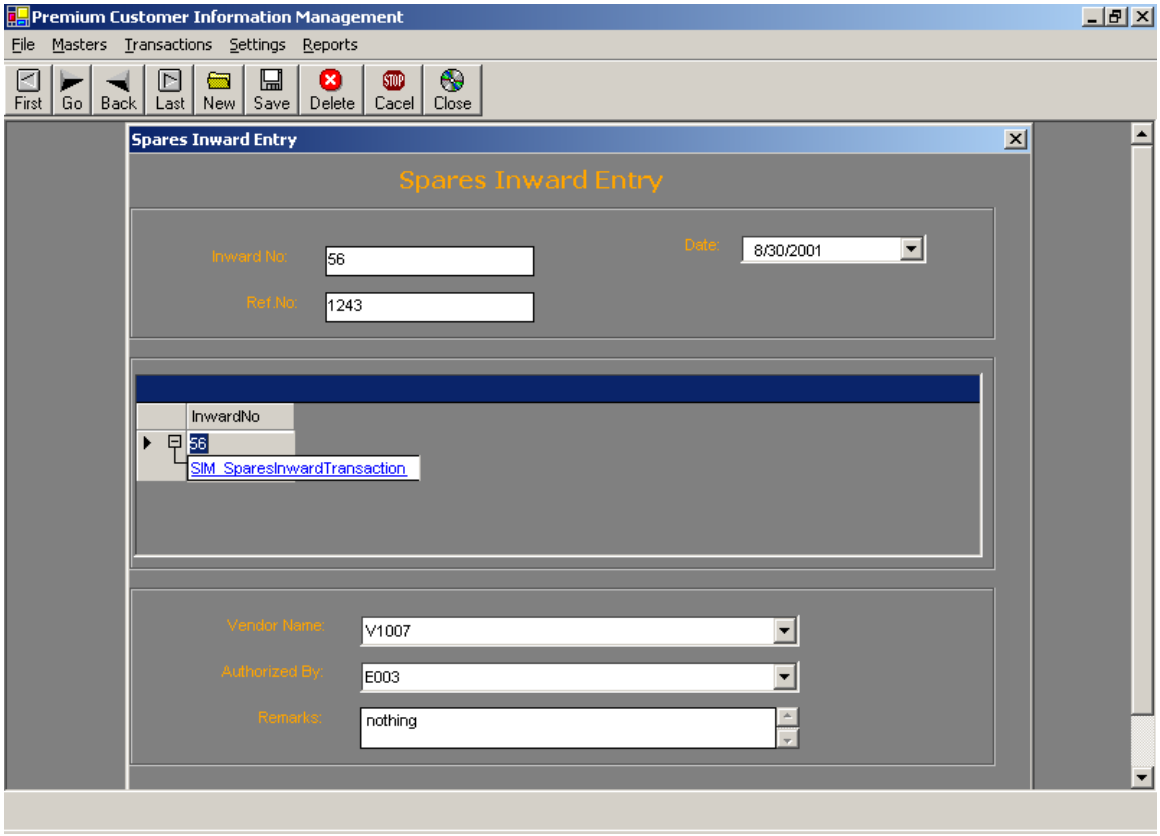


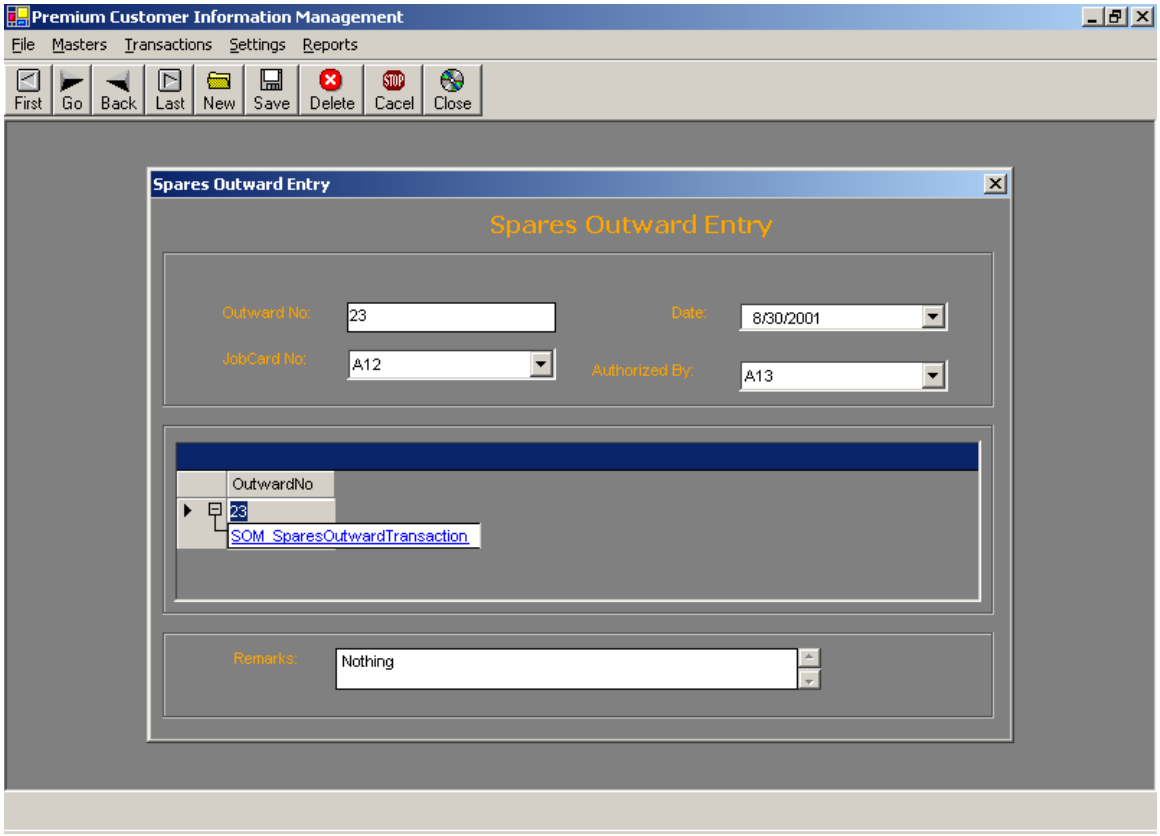
Premium Customer Information Management



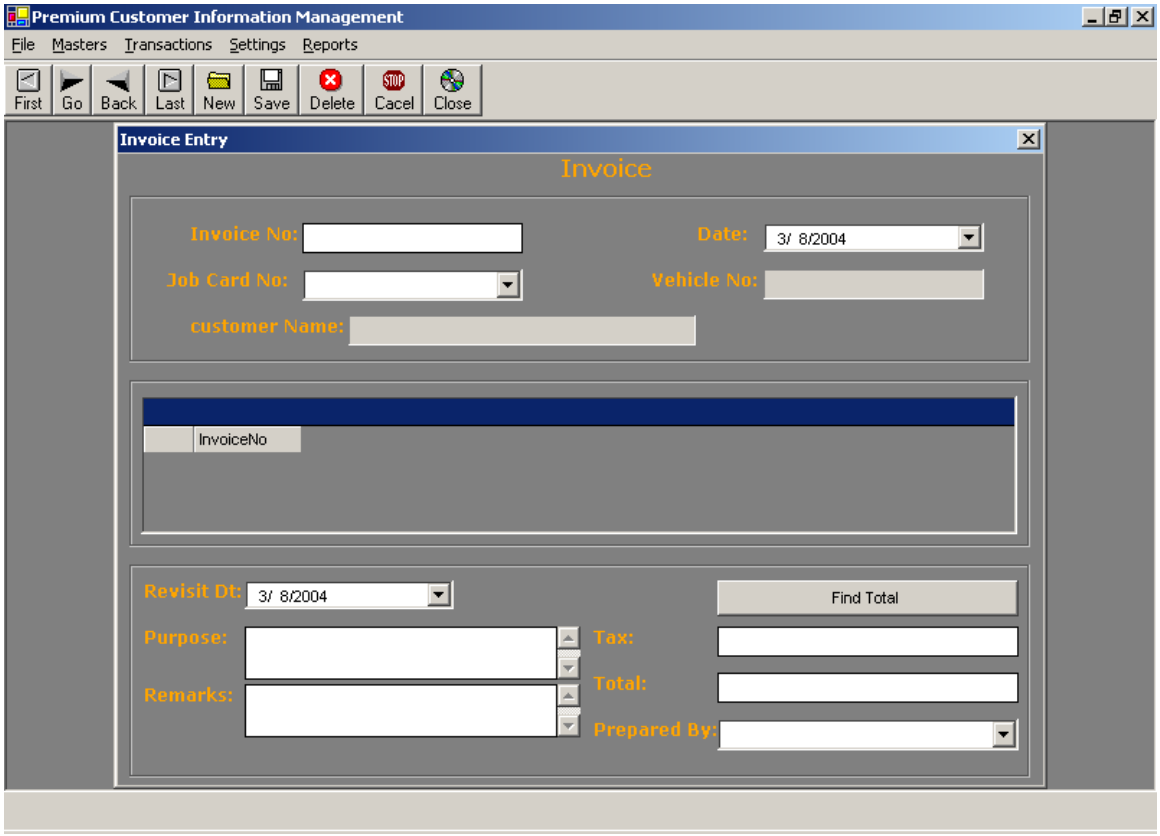
Premium Customer Information Management

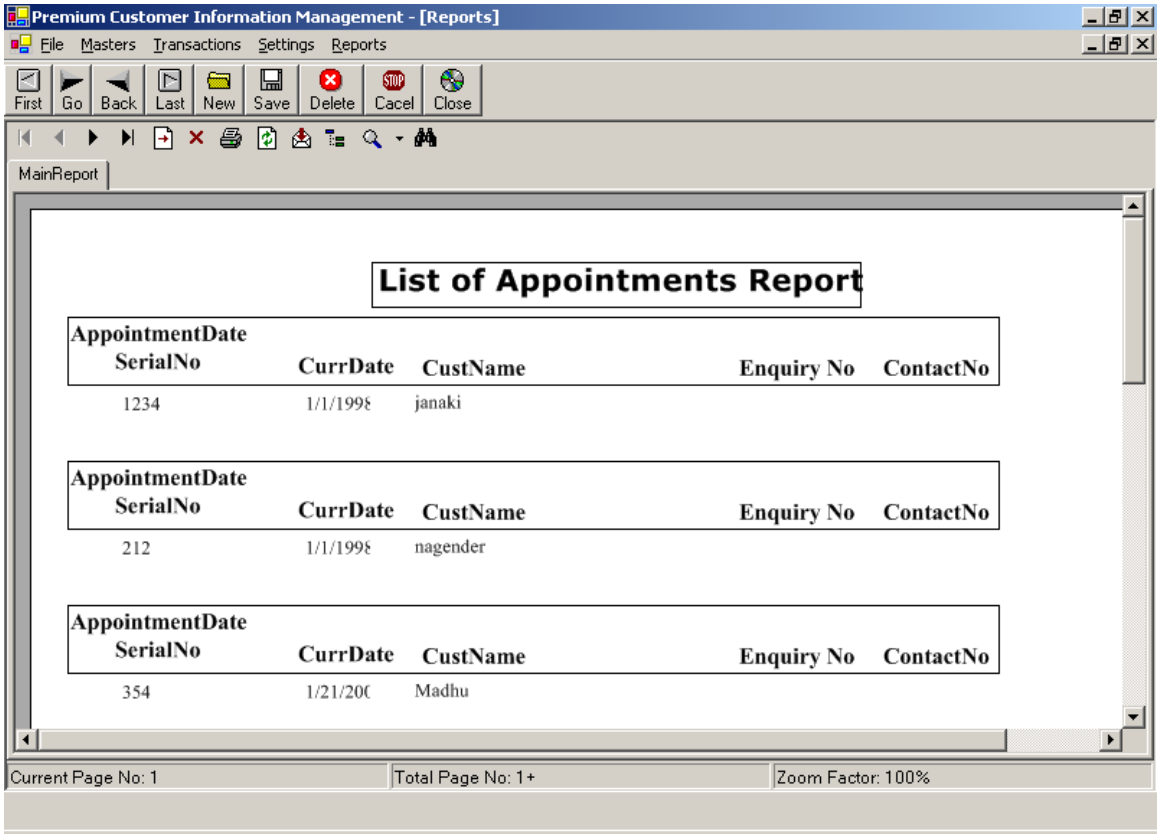


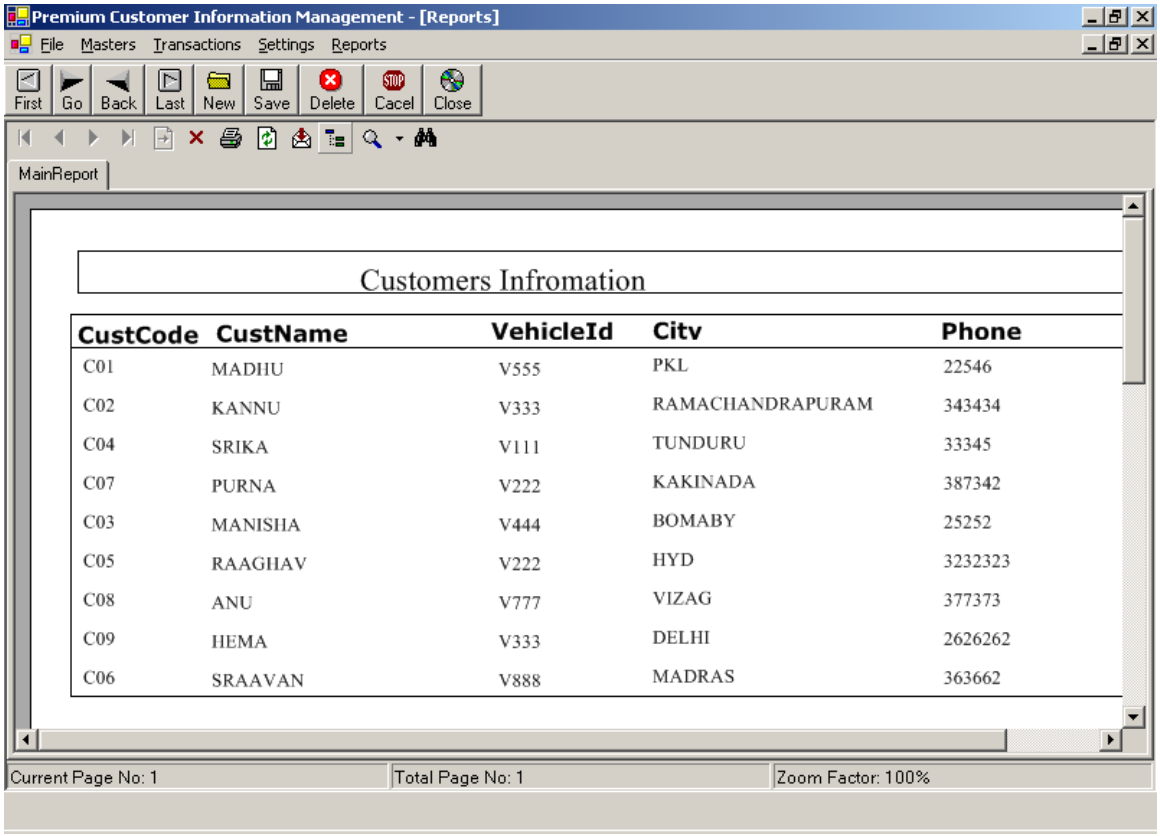

Premium Customer Information Management
File Masters Transactions Settings Reports
First Go Back Last New Save Delete Cacel Close
Spares Inward Entry
Spares Inward Entry
Inward No: 56
Date: 8/30/2001
Ref.No: 1243
InwardNo
56
SIM_SparesInwardTransaction
Vendor Name: V1007
Authorized By: E003
Remarks: nothing

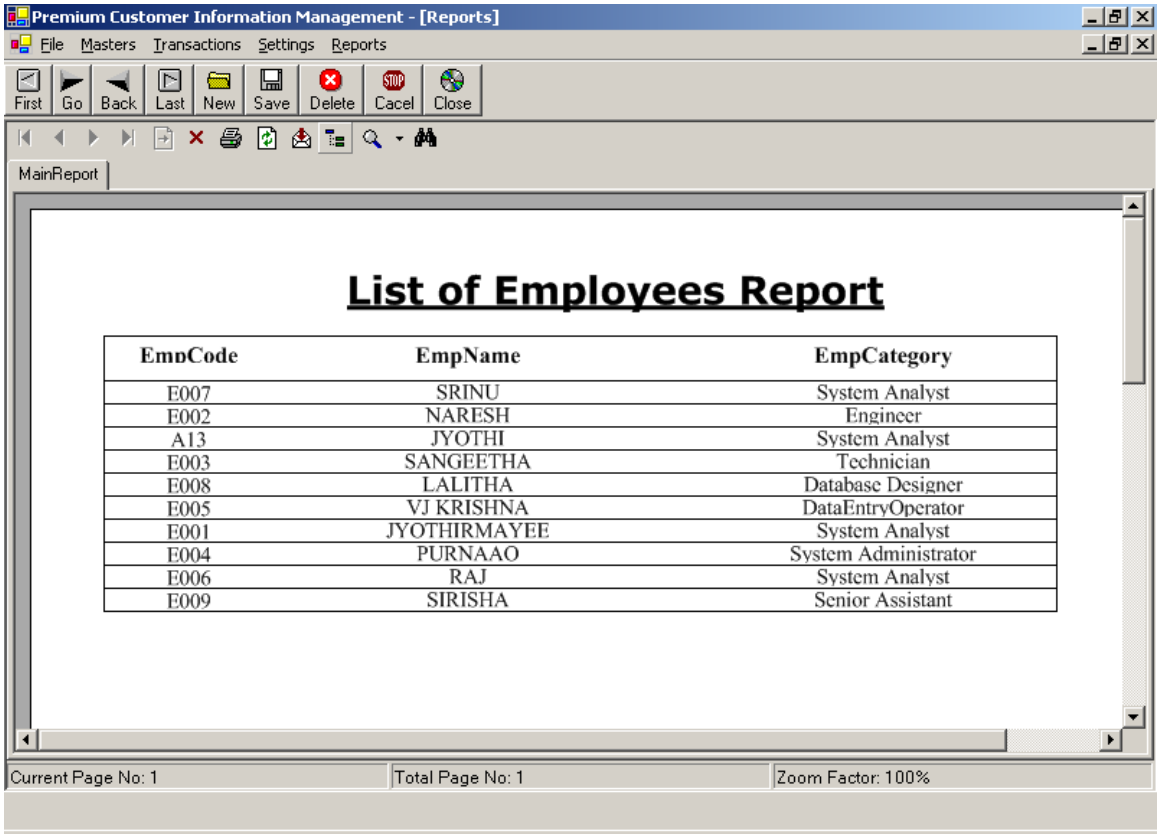

Premium Customer Information Management
File Masters Transactions Settings Reports
First Go Back Last New Save Delete Cacel Close
Spares Outward Entry
Spares Outward Entry
Outward No: 23
Date: 8/30/2001
JobCard No: A12
Authorized By: A13
OutwardNo
23
SOM_SparesOutwardTransaction
Remarks: Nothing

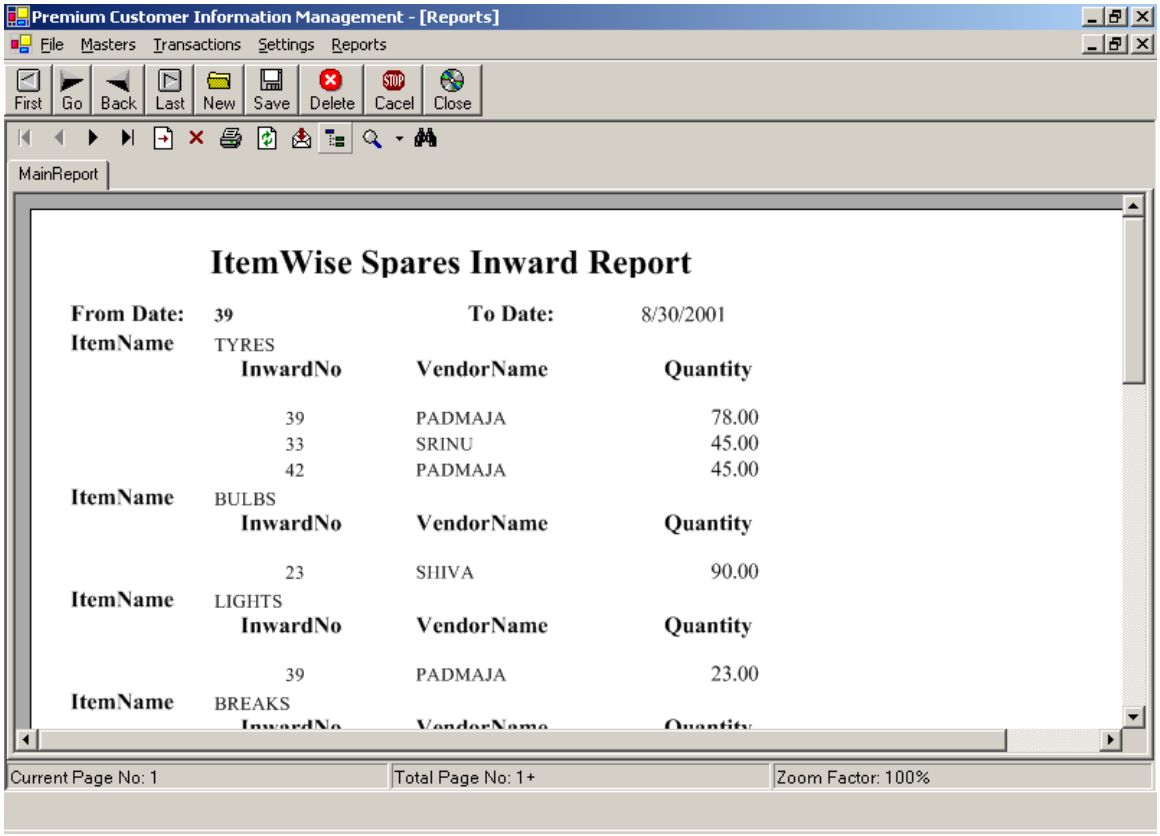
Premium Customer Information Management

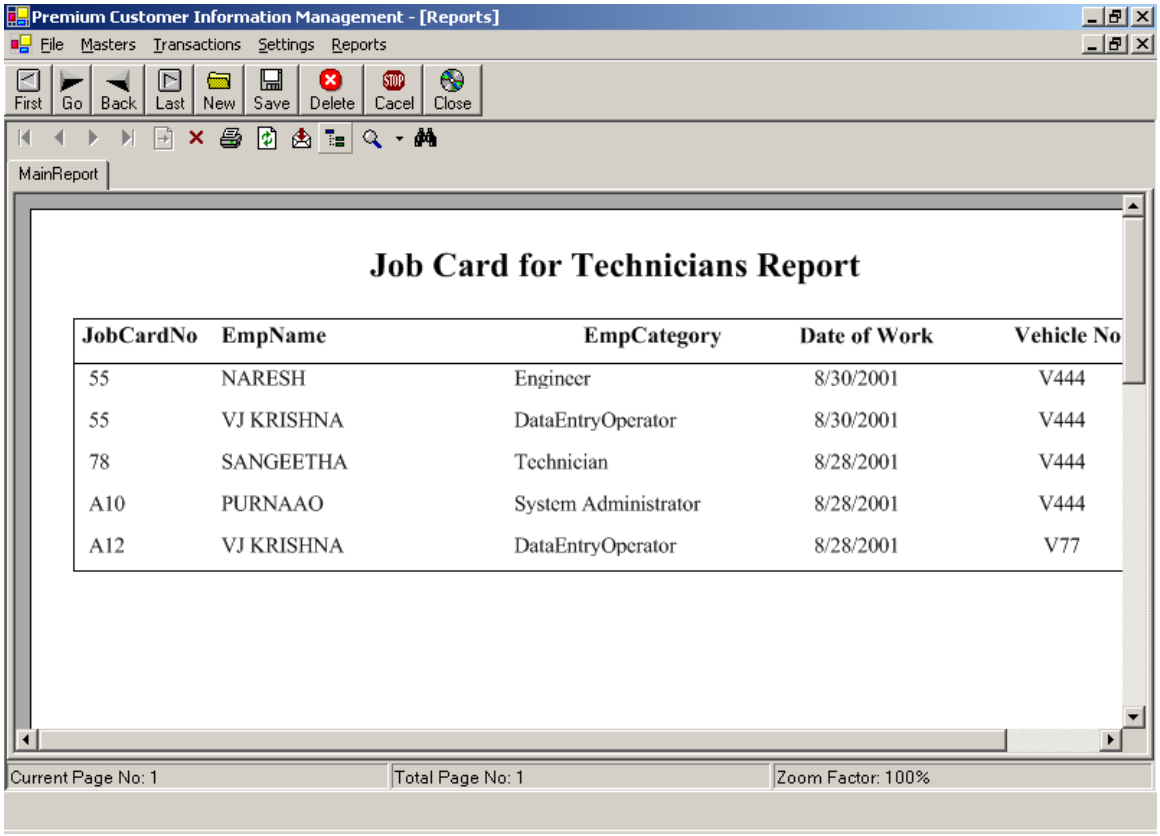


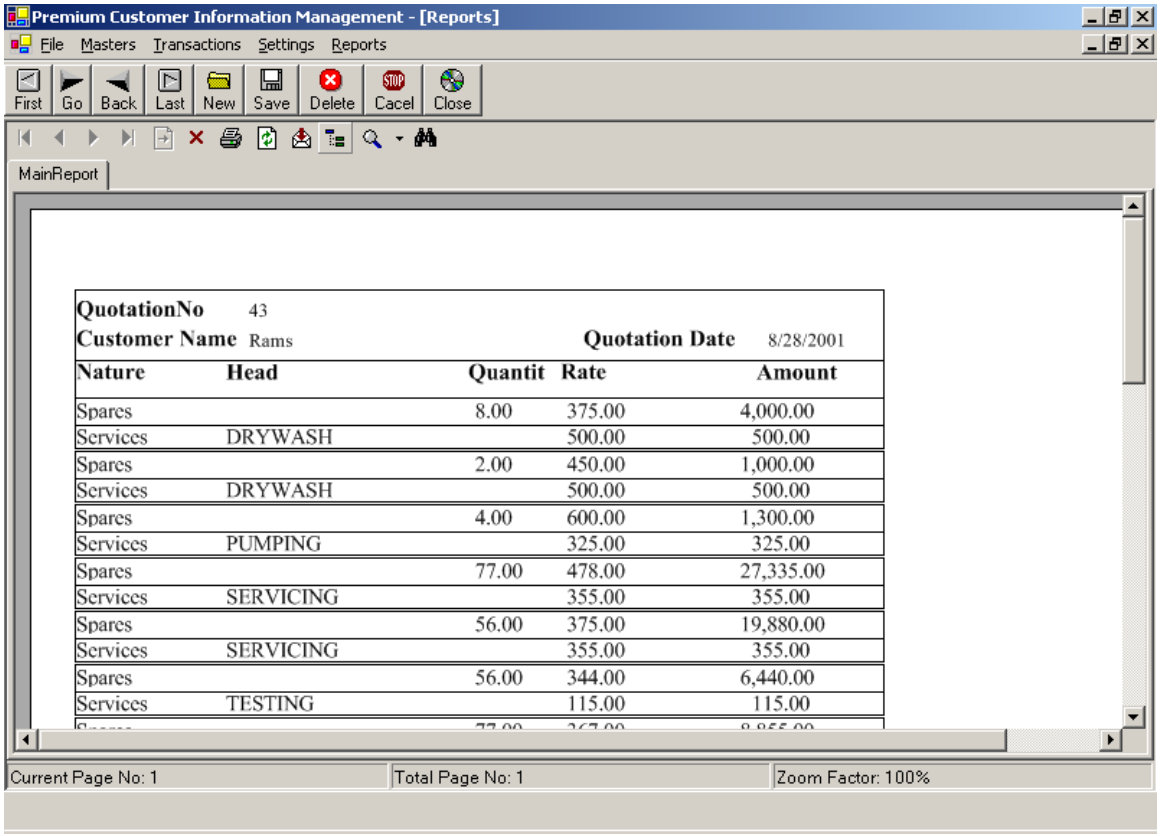

Premium Customer Information Management - [Reports]
File Masters Transactions Settings Reports
First Go Back Last New Save Delete Cacel Close
MainReport
List of Appointments Report
AppointmentDate SerialNo CurrDate CustName Enquiry No ContactNo
1234 1/1/1998 janaki
AppointmentDate SerialNo CurrDate CustName Enquiry No ContactNo
212 1/1/1998 nagender
AppointmentDate SerialNo CurrDate CustName Enquiry No ContactNo
354 1/21/200 Madhu
Current Page No: 1
Total Page No: 1+
Zoom Factor: 100%

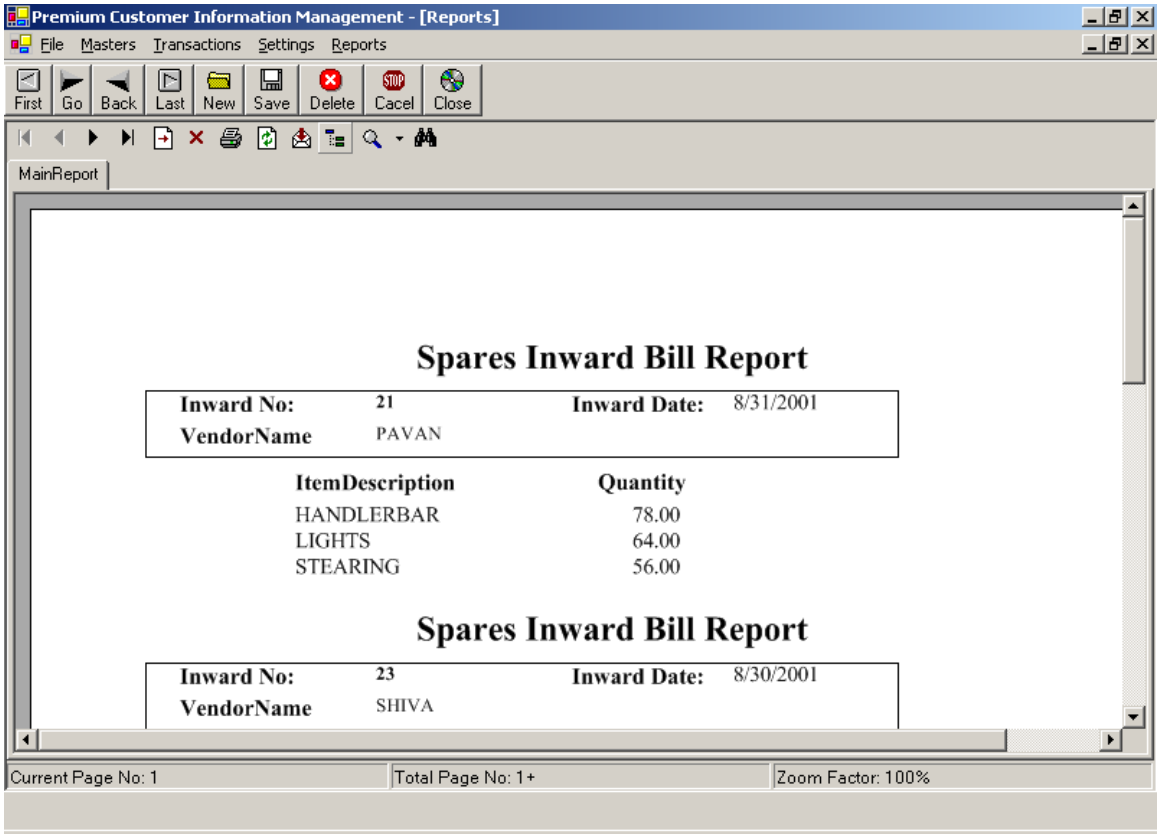

Premium Customer Information Management - [Reports]
File Masters Transactions Settings Reports
First Go Back Last New Save Delete Cacel Close
MainReport
Customers Infromation
CustCode CustName VehicleId Citv Phone
C01 MADHU V555 PKL 22546
C02 KANNU V333 RAMACHANDRAPURAM 343434
C04 SRIKA V111 TUNDURU 33345
C07 PURNA V222 KAKINADA 387342
C03 MANISHA V444 BOMABY 25252
C05 RAAGHAV V222 HYD 3232323
C08 ANU V777 VIZAG 377373
C09 HEMA V333 DELHI 2626262
C06 SRAAVAN V888 MADRAS 363662
Current Page No: 1
Total Page No: 1
Zoom Factor: 100%

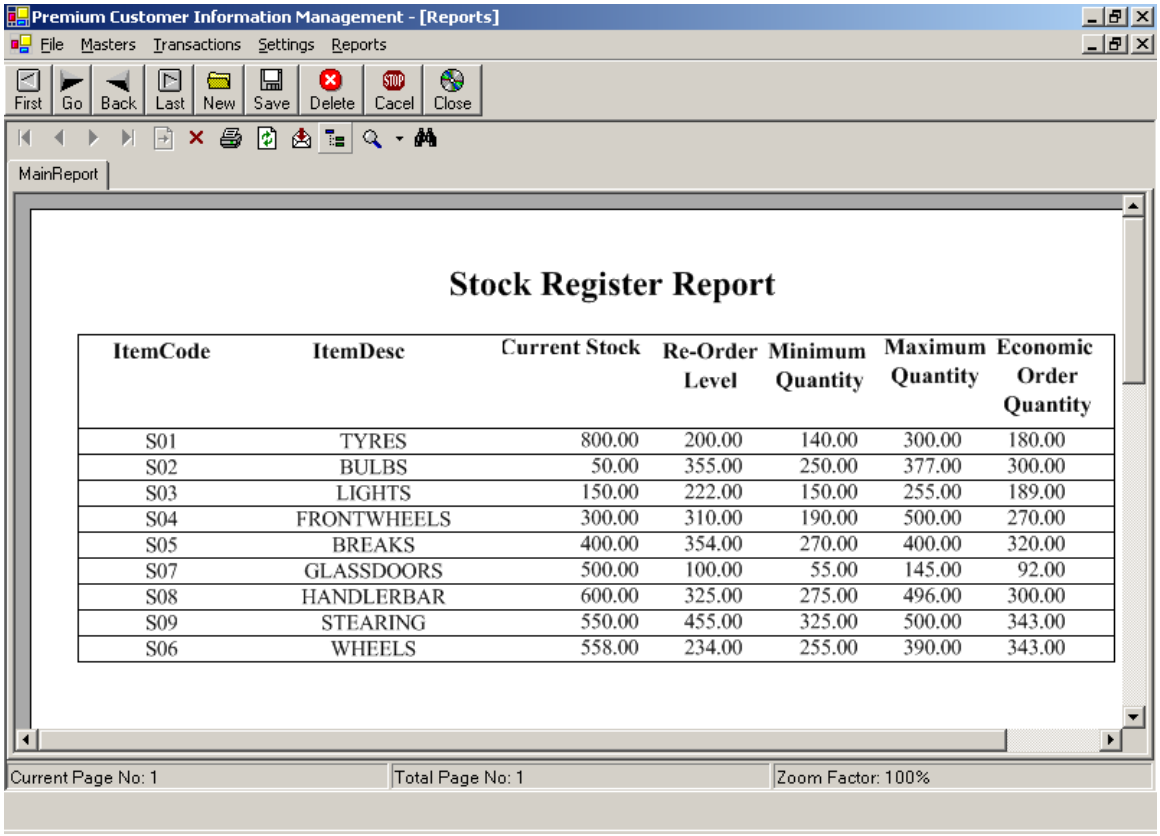

Premium Customer Information Management - [Reports]
File Masters Transactions Settings Reports
First Go Back Last New Save Delete Cacel Close
MainReport
List of Employees Report
EmpCode EmpName EmpCategory
E007 SRINU System Analyst
E002 NARESH Engineer
A13 JYOTHI System Analyst
E003 SANGEETHA Technician
E008 LALITHA Database Designer
E005 VJ KRISHNA DataEntryOperator
E001 JYOTHIRMAYEE System Analyst
E004 PURNAAO System Administrator
E006 RAJ System Analyst
E009 SIRISHA Senior Assistant
Current Page No: 1
Total Page No: 1
Zoom Factor: 100%

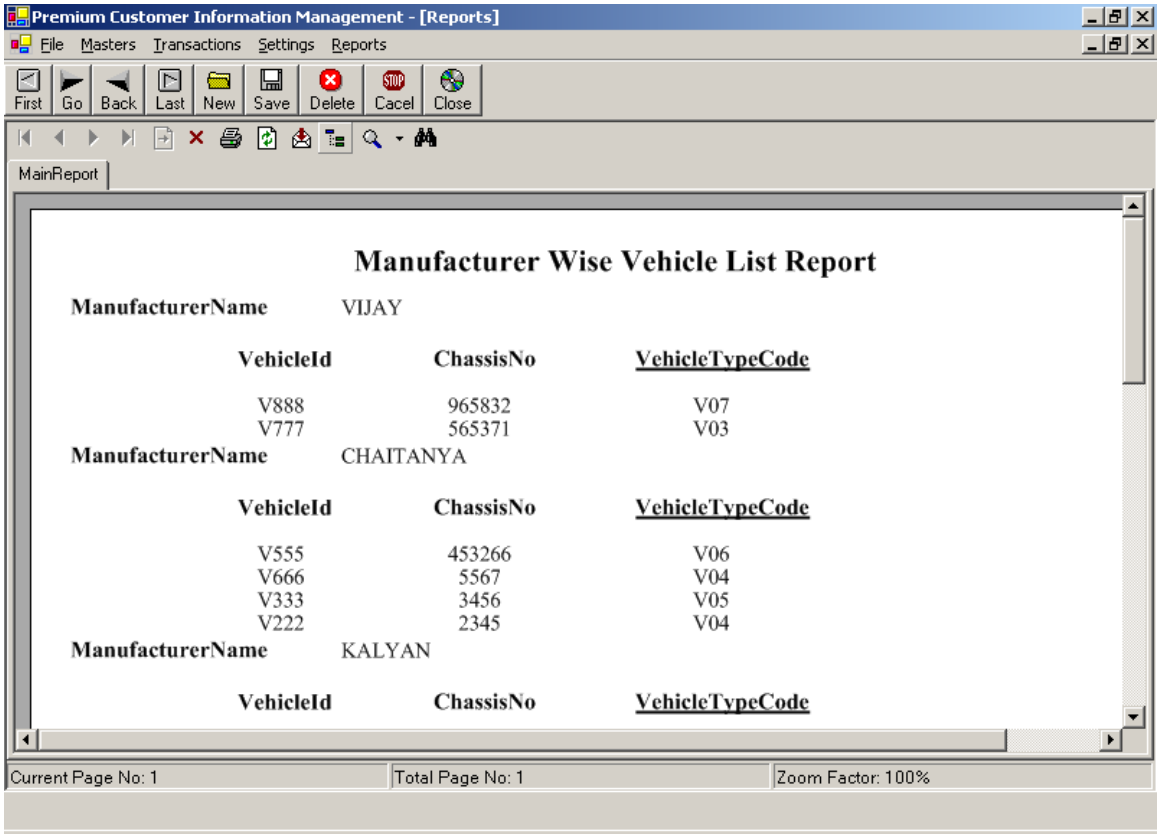

Premium Customer Information Management - [Reports]
File Masters Transactions Settings Reports
First Go Back Last New Save Delete Cacel Close
MainReport
ItemWise Spares Inward Report
From Date: 39 To Date: 8/30/2001
ItemName TYRES
InwardNo VendorName Quantity
39 PADMAJA 78.00
33 SRINU 45.00
42 PADMAJA 45.00
ItemName BULBS
InwardNo VendorName Quantity
23 SHIVA 90.00
ItemName LIGHTS
InwardNo VendorName Quantity
39 PADMAJA 23.00
ItemName BREAKS
Current Page No: 1
Total Page No: 1+
Zoom Factor: 100%

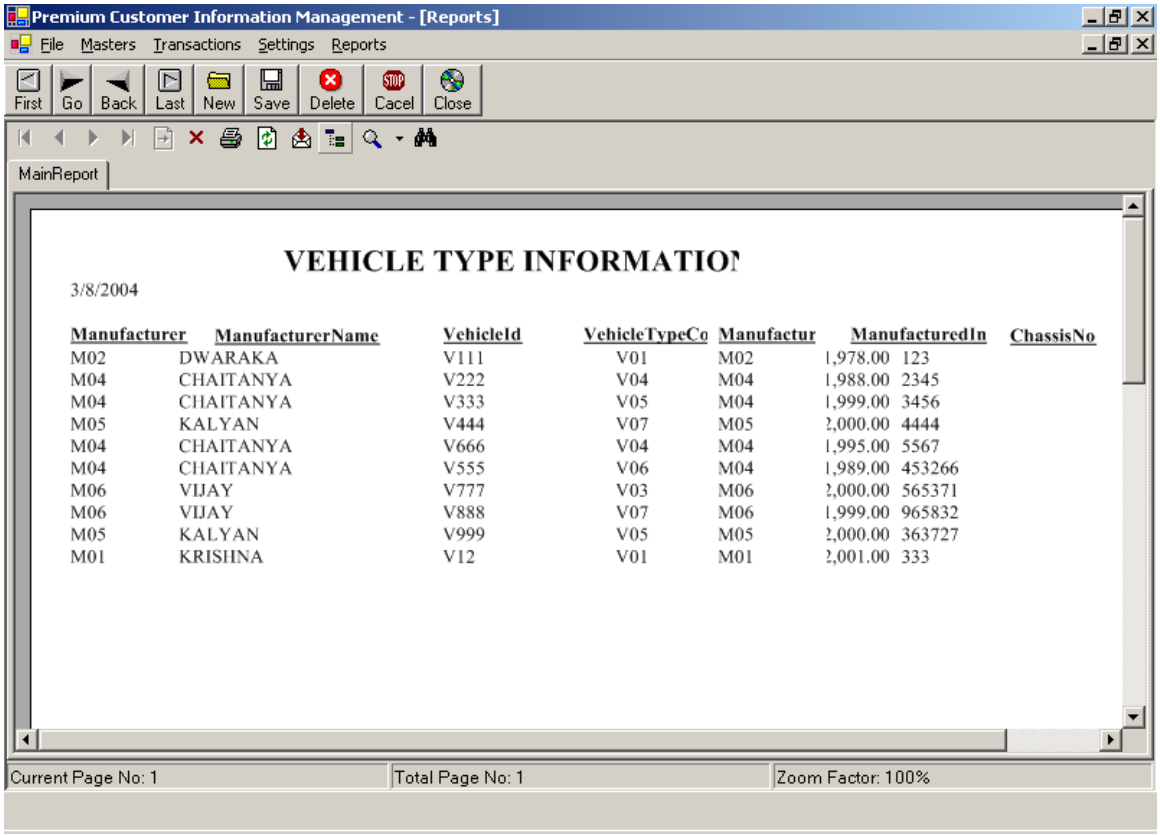

Premium Customer Information Management - [Reports]
File Masters Transactions Settings Reports
First Go Back Last New Save Delete Cacel Close
MainReport
Job Card for Technicians Report
JobCardNo EmpName EmpCategory Date of Work Vehicle No
55 NARESH Engineer 8/30/2001 V444
55 VJ KRISHNA DataEntryOperator 8/30/2001 V444
78 SANGEETHA Technician 8/28/2001 V444
A10 PURNAAO System Administrator 8/28/2001 V444
A12 VJ KRISHNA DataEntryOperator 8/28/2001 V77
Current Page No: 1
Total Page No: 1
Zoom Factor: 100%

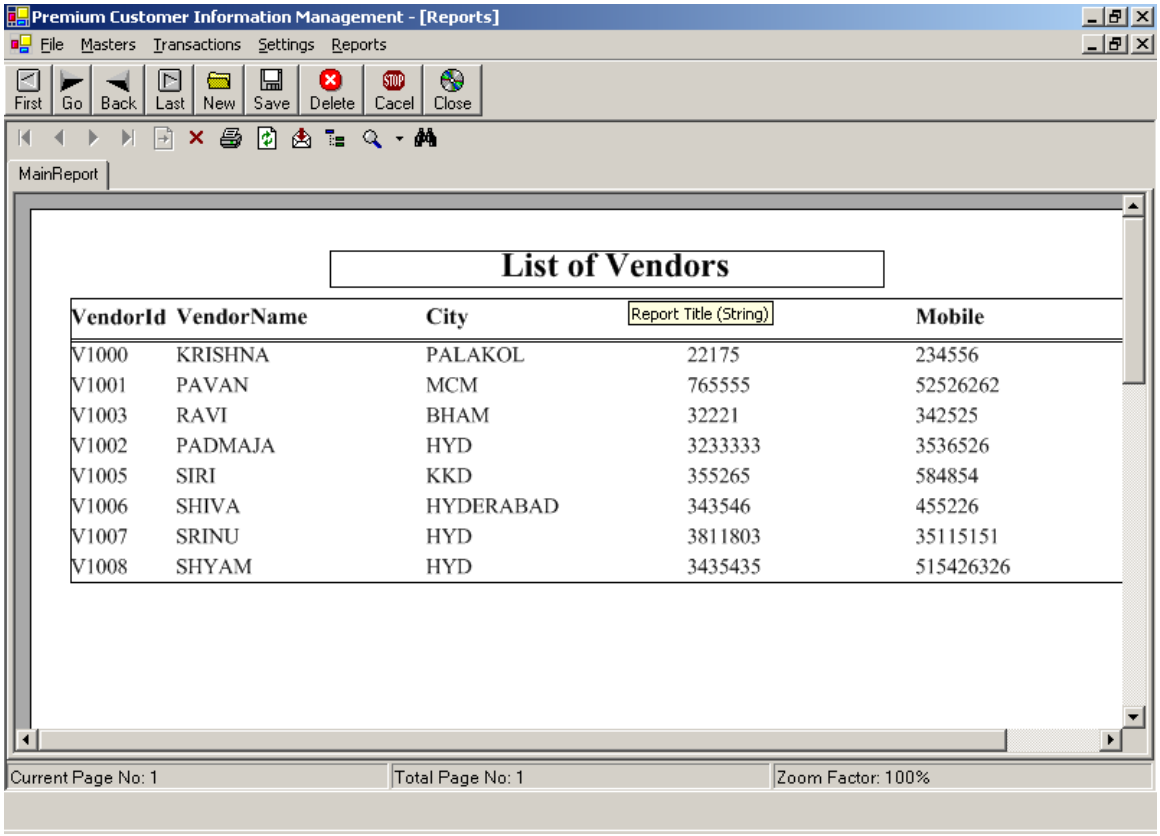

Premium Customer Information Management - [Reports]
File Masters Transactions Settings Reports
First Go Back Last New Save Delete Cacel Close
MainReport
QuotationNo 43
Customer Name Rams
Quotation Date 8/28/2001
Nature Head Quantit Rate Amount
Spares 8.00 375.00 4,000.00
Services DRYWASH 500.00 500.00
Spares 2.00 450.00 1,000.00
Services DRYWASH 500.00 500.00
Spares 4.00 600.00 1,300.00
Services PUMPING 325.00 325.00
Spares 77.00 478.00 27,335.00
Services SERVICING 355.00 355.00
Spares 56.00 375.00 19,880.00
Services SERVICING 355.00 355.00
Spares 56.00 344.00 6,440.00
Services TESTING 115.00 115.00
Current Page No: 1
Total Page No: 1
Zoom Factor: 100%


Premium Customer Information Management - [Reports]
File Masters Transactions Settings Reports
First Go Back Last New Save Delete Cacel Close
MainReport
Spares Inward Bill Report
Inward No: 21 Inward Date: 8/31/2001
VendorName PAVAN
ItemDescription Quantity
HANDLERBAR 78.00
LIGHTS 64.00
STEARING 56.00
Spares Inward Bill Report
Inward No: 23 Inward Date: 8/30/2001
VendorName SHIVA
Current Page No: 1
Total Page No: 1+
Zoom Factor: 100%

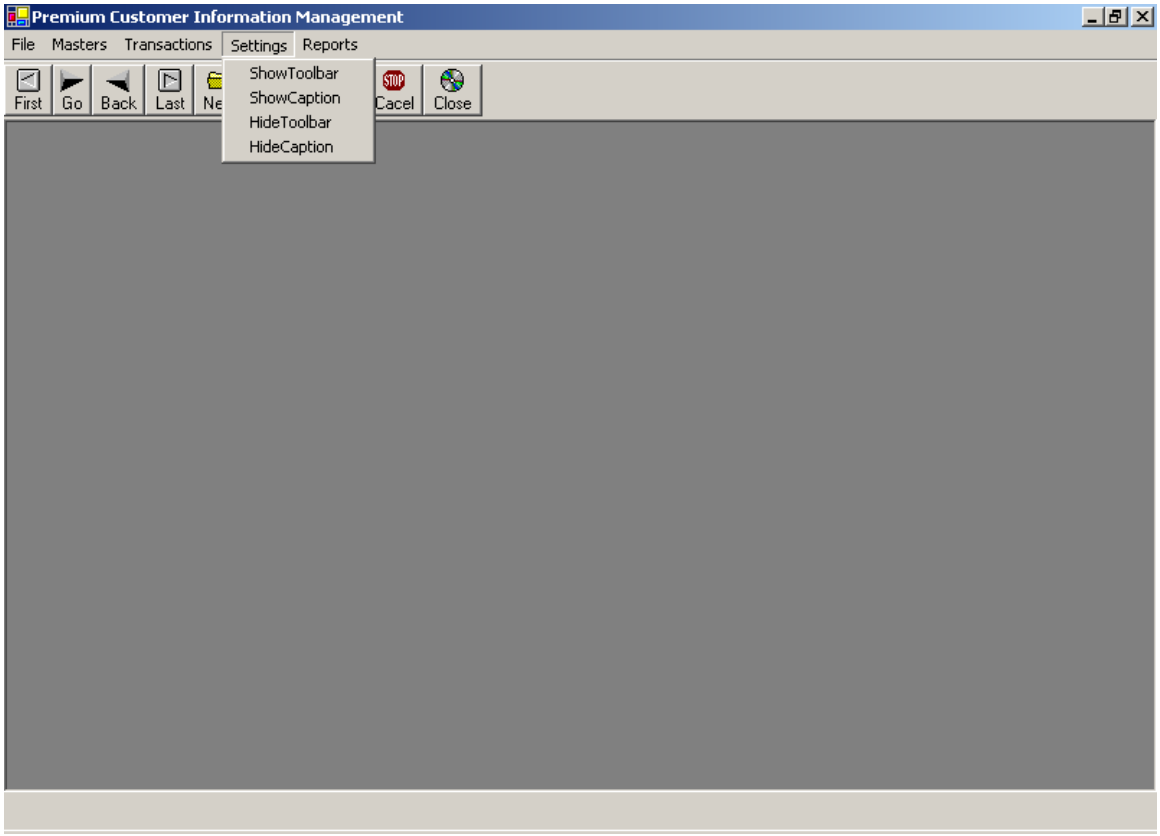

Premium Customer Information Management - [Reports]
File Masters Transactions Settings Reports
First Go Back Last New Save Delete Cacel Close
MainReport
Stock Register Report
ItemCode ItemDesc Current Stock Re-Order Level Minimum Quantity Maximum Quantity Economic Order Quantity
S01 TYRES 800.00 200.00 140.00 300.00 180.00
S02 BULBS 50.00 355.00 250.00 377.00 300.00
S03 LIGHTS 150.00 222.00 150.00 255.00 189.00
S04 FRONTWHEELS 300.00 310.00 190.00 500.00 270.00
S05 BREAKS 400.00 354.00 270.00 400.00 320.00
S07 GLASSDOORS 500.00 100.00 55.00 145.00 92.00
S08 HANDLERBAR 600.00 325.00 275.00 496.00 300.00
S09 STEARING 550.00 455.00 325.00 500.00 343.00
S06 WHEELS 558.00 234.00 255.00 390.00 343.00
Current Page No: 1
Total Page No: 1
Zoom Factor: 100%

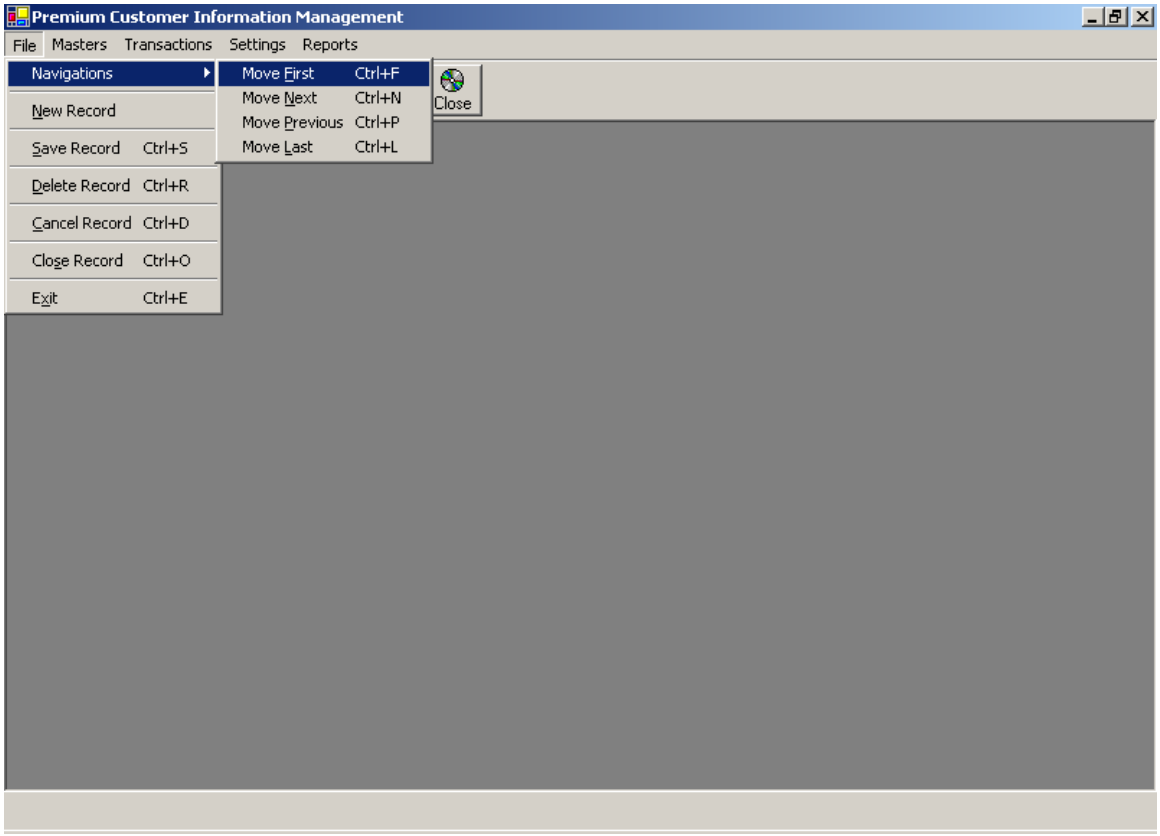

Premium Customer Information Management - [Reports]
File Masters Transactions Settings Reports
First Go Back Last New Save Delete Cacel Close
MainReport
Manufacturer Wise Vehicle List Report
ManufacturerName VIJAY
VehicleId ChassisNo VehicleTypeCode
V888 965832 V07
V777 565371 V03
ManufacturerName CHAITANYA
VehicleId ChassisNo VehicleTypeCode
V555 453266 V06
V666 5567 V04
V333 3456 V05
V222 2345 V04
ManufacturerName KALYAN
VehicleId ChassisNo VehicleTypeCode
Current Page No: 1
Total Page No: 1
Zoom Factor: 100%


Premium Customer Information Management - [Reports]
File Masters Transactions Settings Reports
First Go Back Last New Save Delete Cacel Close
MainReport
VEHICLE TYPE INFORMATIO
3/8/2004
Manufacturer ManufacturerName VehicleId VehicleTypeCo Manufactur ManufacturedIn ChassisNo
M02 DWARAKA V111 V01 M02 1,978.00 123
M04 CHAITANYA V222 V04 M04 1,988.00 2345
M04 CHAITANYA V333 V05 M04 1,999.00 3456
M05 KALYAN V444 V07 M05 2,000.00 4444
M04 CHAITANYA V666 V04 M04 1,995.00 5567
M04 CHAITANYA V555 V06 M04 1,989.00 453266
M06 VIJAY V777 V03 M06 2,000.00 565371
M06 VIJAY V888 V07 M06 1,999.00 965832
M05 KALYAN V999 V05 M05 2,000.00 363727
M01 KRISHNA V12 V01 M01 2,001.00 333
Current Page No: 1
Total Page No: 1
Zoom Factor: 100%


Premium Customer Information Management - [Reports]
File Masters Transactions Settings Reports
First Go Back Last New Save Delete Cacel Close
MainReport
List of Vendors
Report Title (String)
VendorId VendorName City Mobile
V1000 KRISHNA PALAKOL 22175 234556
V1001 PAVAN MCM 765555 52526262
V1003 RAVI BHAM 32221 342525
V1002 PADMAJA HYD 3233333 3536526
V1005 SIRI KKD 355265 584854
V1006 SHIVA HYDERABAD 343546 455226
V1007 SRINU HYD 3811803 35115151
V1008 SHYAM HYD 3435435 515426326
Current Page No: 1
Total Page No: 1
Zoom Factor: 100%

Premium Customer Information Management





System Testing

4.1 Test Plan

The importance of software testing and its implications cannot be overemphasized. Software testing is a critical element of Software Quality Assurance and represents the ultimate review of the specifications, design and coding.

4.1.1. Testing Objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

- Testing is a process of executing a program with the intent of finding an error.
- A successful test is one that uncovers an as yet undiscovered error.
- A good test case is one that has a high probability of finding error, if it exists.

But there is one thing that testing cannot do (just to quote a very famous sentence) "*Testing cannot show the absence of defects, it can only show that software defects are presents*."

As the test results are gathered and evaluated they begin to give a qualitative indication of the reliability of the software. If severe errors are detected, the overall quality of the software is a natural suspect. If, on the other hand, all the errors, which are encountered, are easily modifiable, then one of the two conclusions can be made:

- The tests are inadequate to detect possibly present errors.
- The software more or less confirms to the quality and reliable standards.

For the purpose of the current project we are assuming that in the event that errors that are easily modifiable points to the latter possibility, since repeating the entire testing routine can be very time consuming. What we propose to do instead is to get it tested by one or more persons who are not a part of the development team but is well versed with the subject and with the concept of software testing (alpha testing). If he can detect no serious errors, it will enable us to state with more confidence that the software does actually conform to the expected standards.

4.1.2. **Testing Strategy**:

A testing strategy is a roadway, giving there how-to conducting a test. Our testing strategy is flexible enough to promote customization that may be necessary in due course of development process. For instance during coding we find that a change in design (e.g. Z denormalized table makes certain query easy to process), we maintain a change log and refer to it at appropriate time during the testing. Software can be tested in one of the following ways:

- Knowing the specific functions that the Software is expected to perform, tests can be conducted to perform that all functions are fully operational.
- Knowing the internal workings of the product, tests can be conducted to show that internal operations of the system perform according to the specifications and all internal components are adequately exercised.

The first approach is what is known as Black box testing and the second is called White box testing. We will be using a mixed approach, more popularly known as sandwich testing. We apply white box testing techniques to ascertain the functionalities top-down and then we use black box testing to demonstrate that everything runs as expected.

4.1.2.1. Unit testing:

Unit testing focuses verification effort on the smallest unit of software i.e. the module. Using the detailed design and the process specifications testing is done to uncover errors within the boundary of the module. All nodules must be successful in the unit test before the start of the integration testing begins.

4.1.2.2. Integration Testing:

Integration testing is a systematic technique for constructing the program structure while conducting tests at the same time to uncover errors associated with interfacing. We have used incremental integration testing for this project.

4.1.2.3. Validation Testing:

At the culmination of integration testing the software is complete as a package and the interfacing errors have been uncovered and fixed, final tests- validation testing- may begin. Validation tests succeed when the software performs exactly in the manner as expected by the user.

Software validation is done by a series of Black box tests that demonstrate the conformance with requirements. Alpha and bets testing fall in this category. We will not do beta testing but alpha testing will certainly will certainly be done.

4.2 Test Case Design

The techniques that are used in deriving the test cases are explained below.

4.2.1. Condition Testing

Condition Testing is a test case design method that exercises the logical conditions contained in the program. The possible components in a condition statement are: a Boolean operator, a Boolean variable, a relational operator, arithmetic expression and parenthesis around simple or compound conditions. The condition testing method focuses on testing each condition in the program.

It has two advantages over the pure white box testing:

- Measurement of test coverage is straightforward.
- The test coverage of conditions in a program provides guidance for generation of additional tests for the program.
- We have used a special condition testing method called the Domain Testing where the condition in a form lie E1 < Relational Operator> E2 will the tested for three conditions E1 > E2, E1 equal to E2 and E1 les than E2.

4.2.2. Boundary Value Analysis:

Boundary value analysis leads to a selection of test cases that exercise the boundary conditions or bounding values. It has been observed that a large number of errors tend to appear at the boundaries of the input domain than in the center. The guidelines for developing the test cases in BVA are given below:

- If the input condition has a low and high range then tests should be done at the low and high boundaries. Values above and below these extremes should be tested.
- Apply the same principle to output conditions. E.g.: Test cases should be designed to generate the maximum and minimum no of entries in a report that is generated by the program.

4.2.3. Equivalence Partitioning:

Equivalence partitioning is a black box testing method that divides the input domain of a program into classes of data from which test cases can be derived. A typical test case uncovers a class of errors that might otherwise require many more test cases before the error is observed.

Equivalence classes for input determine the valid and invalid inputs for the program. Equivalence class test cases are generated using the following guidelines:

- If an input class specifies a range then one valid and two invalid equivalence classes are defined.
- If an input class specifies a value then one valid and one invalid equivalence classes are defined.
- If an input class specifies a member of a set then one valid and one invalid equivalence classes are defined.
- If an input class specifies a Boolean then one valid and two invalid equivalence classes are defined.

Test cases should be selected so that the largest number of attributes of an equivalence class is exercised at once.

Implementation

5.1 Change Over

Change over is a process of converting from one to another. The new system may involve installing new hardware and preparing data. There are four methods of handling systems conversion.

5.1.1 Direct Changeover:

This method is a complete replacement of the old system by the new system, in one move. It is a bold move, which should be undertaken only when every one concerned has confidence on the new system. This method is potentially the least expensive but the most risky.

5.1.2. Parallel Running:

Parallel running or operation means processing current data by new system to crosscheck the results with the old system. Its main attraction is that the old system is kept alive and operational until the new system has been proved as a success one. Its main disadvantage is extra cost.

5.1.3. Pilot Running:

Pilot Running is similar to the concept of Parallel running. This method is more like an extended system test, but it may be considered a more practical form of change over for organizational reasons.

5.1.4. Staged Changeover:

Staged changeover involves a series of limited size direct changeovers, the new system being introduced piece-by-piece. Its disadvantages are that it creates problems of controlling the selected parts of the old and new system and it tends to prolong the implementation period.

The changeover planned to Aqua Gold is a parallel running changeover method from old system to the new system. The old system is kept alive and operational until the new system has been proved for at least one system cycle, using full live data in the real operational environment of place, compared with the old system before acceptance by the user thus by promoting user confidence.

5.2. Education and Training

Education involves creating the right atmosphere and motivating user staff. The sessions are targeted to make the computer system more acceptable by reasoning out and explaining the benefits this package could provide to the user and the amount of clerical effort that is being reduced. Also the users are educated how their work would get interesting rather than feeling this would eliminate their jobs. The existing jobs are changed but not eliminated.

Training sessions are targeted on how to work with the package, how to navigate through the package and so on. Training sessions are felt essential in the client's work place, as it would help them to work with the package in a faster and in an efficient manner.

Concluding Remarks

The customized package "Aqua Gold 1.0" fulfills the requirements as stated by the client. This package took the first step in automizing the client's manual system but a scope of improvement is a part of life and is inevitable for survival and this is no exception to this package. Any improvements if found necessary would be incorporated in the next release of Aqua Gold.